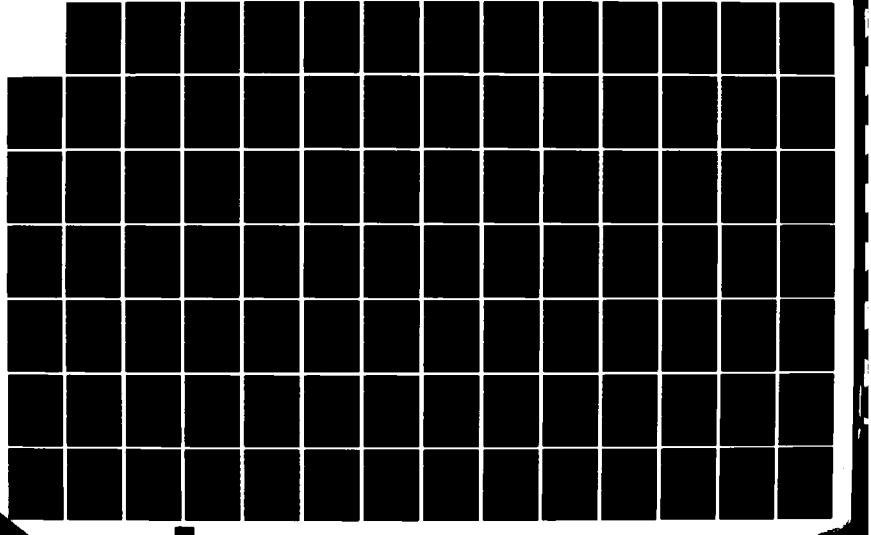
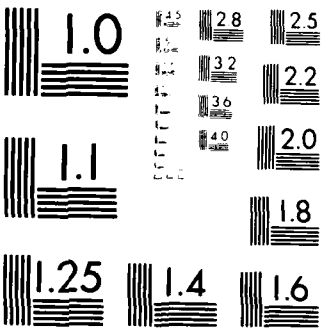
AD-A152 134 AN ANALYSIS OF DATA DICTIONARIES AND THEIR ROLE IN INFORMATION RESOURCE MANAGEMENT(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA S L LANDIN ET AL. SEP 84 1/2

UNCLASSIFIED F/G 5/2 NL





(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

AN ANALYSIS OF DATA DICTIONARIES
AND THEIR ROLE IN
INFORMATION RESOURCE MANAGEMENT

by

Suzanne L. Landin
Ronald L. Owens

September 1984

Thesis Advisor: Daniel R. Dolk

Approved for public release; distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A152134	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) An Analysis of Data Dictionaries and Their Role in Information Resource Management		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Suzanne L. Landin Ronald L. Owens		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943		12. REPORT DATE September 1984
		13. NUMBER OF PAGES 108
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Data Dictionary; Distributed Data Processing; Information Resource Management; Federal Information Processing Standard (FIPS) for Data Dictionary Systems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The goal of efficient management of an organization's information resource can be accomplished through the implementation and use of a data dictionary. This thesis defines the structure and functions of a data dictionary and analyzes the attempt of the National Bureau of Standards to promulgate a standard software specification for use in the evaluation and selection of data dictionaries in the federal government. Criteria for the "ideal"

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

data dictionary are developed based on the role a dictionary can play in information resource management and are then used to evaluate four commercial data dictionary packages. Finally, some ideas concerning possible applications for data dictionary technology are presented.

Approved for public release; distribution unlimited.

An Analysis of Data Dictionaries and Their Role in Information Resource Management

by

Suzanne L. Landin
Lieutenant, United States Navy
B.A., University of Washington, 1978

and

Ronald L. Owens
Lieutenant Commander, United States Navy
B.B.A., Hardin-Simmons University, 1971

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 1984

Authors: Suzanne L. Landin
Suzanne L. Landin

Ronald L. Owens
Ronald L. Owens

Approved by: Daniel R. Dolk
Daniel R. Dolk, Thesis Advisor

F. Marchman Perry
F. Marchman Perry, Second Reader

Willis R. Greer, Jr., Chairman,
Department of Administrative Sciences

Kneale T. Marshall
Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

The goal of efficient management of an organization's information resource can be accomplished through the implementation and use of a data dictionary. This thesis defines the structure and functions of a data dictionary and analyzes the attempt of the National Bureau of Standards to promulgate a standard software specification for use in the evaluation and selection of data dictionaries in the federal government. Criteria for the "ideal" data dictionary are developed based on the role a dictionary can play in information resource management and are then used to evaluate four commercial data dictionary packages. Finally, some ideas concerning possible applications for data dictionary technology are presented.

TABLE OF CONTENTS

I. THE NEED FOR A DATA DICTIONARY 9

A. BACKGROUND 9

B. PURPOSE OF THE THESIS 11

II. THE ANATOMY OF A DATA DICTIONARY 12

A. INTRODUCTION 12

B. THE STRUCTURE OF A DATA DICTIONARY 16

C. THE FUNCTIONS OF A DATA DICTIONARY 20

1. Definition 20

2. Update 21

3. Retrieval 21

4. Software Interface 21

III. FEDERAL INFORMATION PROCESSING STANDARD FOR DATA DICTIONARY SYSTEMS 23

A. INTRODUCTION 23

B. SYSTEM STANDARD SCHEMA 24

C. COMMAND LANGUAGE INTERFACE SPECIFICATIONS . . 25

D. INTERACTIVE INTERFACE SPECIFICATIONS 26

E. DICTIONARY ADMINISTRATOR SUPPORT SPECIFICATIONS 27

F. EVALUATION 28

IV. THE ROLE OF THE DATA DICTIONARY IN INFORMATION RESOURCE MANAGEMENT 31

A. INFORMATION RESOURCE MANAGEMENT 31

1. Planning Phase 33

2. Study Phase 34

3. Design/Coding Phase 35

4. Operation and Maintenance 35

B. OBJECTIVES OF A DATA DICTIONARY 37
1. Data Security 37
2. Data Integrity 38
3. Documentation/Maintenance 40
C. THE IDEAL DATA DICTIONARY 41
1. System Standard Schema and Extensibility 42
2. Command and Query Languages 43
3. Ease of Use 44
4. Security 44
5. Documentation and Reports 44
6. Application Interfaces 45

V. EVALUATION OF COMMERCIAL DATA DICTIONARIES 46
A. DATA DESIGNER 46
B. MSP DATAMANAGER 54
C. ADR DATADICTIONARY 65
D. ORACLE 76
E. COMPARISON OF DATA DESIGNER, DATAMANAGER, DATADICTIONARY, AND ORACLE 83

VI. EXPANSIONS OF THE ROLE OF DATA DICTIONARIES . . . 90
A. DISTRIBUTED DATA PROCESSING 90
B. DECISION-MAKING 97
1. The Decision-Making Process 98
2. Crisis Management 100
C. CONCLUSIONS 101

APPENDIX A: BACKUS-NAUR FORM 104

LIST OF REFERENCES 105

INITIAL DISTRIBUTION LIST 109

LIST OF TABLES

1. Standard Commands of DATA DESIGNER 47
2. DATA DESIGNER Modeling Codes 50
3. DATA DESIGNER Generate Options 52
4. Reports Available with DATA DESIGNER 52
5. DATAMANAGER Standard Commands 57
6. DATAMANAGER Standard Schema Descriptors 57
7. DATAMANAGER Maintenance Commands 64
8. DATAMANAGER Report/Query Commands 64
9. Components of ADR's DATCOM System 67
10. ADR DATADICTIONARY Standard Entity-types 68
11. Principal Reports of DATADICTIONARY 75
12. Category One: Schemas and Extensibility 84
13. Category Two: Command/Query Languages 85
14. Category Three: Relative Ease of Use 86
15. Category Four: Security 86
16. Category Five: Documentation and Reports 87
17. Category Six: Application Interfaces 87
18. Data Dictionary Comparison Totals 88

LIST OF FIGURES

2.1	Types of Data Dictionary Metadata	13
2.2	Free-standing and Dependent Data Dictionaries	14
2.3	Views Within a DBMS	16
2.4	Sample Schema Descriptors	18
2.5	Comparison of Data Levels	20
5.1	DATAMANAGER's Hierarchy of Entity-types	58
5.2	STUDENT example in DATAMANAGER Structure	62
5.3	A Logical Hierarchy of Entity-types	69
5.4	ADR DATADICTIONARY Master Menu	71
5.5	ORACLE's Logical Hierarchy	77
5.6	Tables of the ORACLE Data Dictionary	78
5.7	ORACLE CATALOG Listing	79
5.8	ORACLE SYSCATALOG Listing	79
5.9	ORACLE CATALOG Listing With New View	80
5.10	ORACLE SYSTABAUTH Listing for User Owens	81
6.1	Duplicated Data Dictionaries	94
6.2	Partitioned Data Dictionary (DD)	95
6.3	Hierarchy of Distributed Data Dictionaries	96

I. THE NEED FOR A DATA DICTIONARY

A. BACKGROUND

One of the most important resources of an organization and one that is too often overlooked is data. People, dollars, materials, and time are usually well controlled and budgeted, yet the data about an organization and its operations is often managed haphazardly, if at all.

Database technology has made possible the storage and processing of an organization's data as an integrated whole and allows the sharing of that processed data, or information, throughout the organization. A database management system (DBMS) acts as a librarian for the database, storing and retrieving data according to a particular format [Ref. 1]. However, a DBMS does not necessarily provide for the security, integrity, accountability, or maintainability of data. These objectives are best achieved when a data dictionary is used in conjunction with the DBMS.

Simply stated, a data dictionary is a central repository of descriptive data about the definition, characteristics, location, and usage of the data found in an organization. A fully utilized data dictionary will control the collection, maintenance, and retrieval of this data. For example, if the aircraft carrier U.S.S. Constellation had a data dictionary, it would be possible to ask questions such as

What type of data is contained in a "Controlled Equipage" record?

How many programs use the "Personnel" file?

Which departments receive the "Ammunition Transaction" report?

What is the relationship between "Inventory Item" and "Reorder Point"?

In which records is the field "Social Security Number" found?

Who is authorized to update the "Readiness Status" field?

What is the range of values for "Readiness Status" data?

In which database is the "Preventive Maintenance" file found?

Those who will benefit from the answers to these questions include not only the ship's data administrator, but also programmers, systems development personnel, data processing staff, auditors, and, most important, end users at every level of the organization.

Even though data dictionary software has been available commercially since 1970 and the advantages and benefits associated with data dictionaries are widely recognized, most organizations have been slow to implement them, and the Department of Defense is no exception. A recent study by the Committee on Review of Navy Long-Range Automatic Data Processing Planning [Ref. 2] points out that

> Virtually every action by a commander, manager, or administrator in the Navy, as in any large organization, involves the acquisition and understanding of information: information about the organization, about its status, about its resources, about its environment. His actions usually result in the creation and promulgation of policies and directives: that is, information for subordinates, peers, or superiors.

If it is true that "the benefit derived from a dictionary is proportional to the size of the dictionary itself," [Ref. 3] the military stands to gain a great deal from the implementation of data dictionaries.

At present, there is no consensus in computing literature about exactly what a data dictionary should do or what kind of data dictionary is best for a particular organization. There are many different data dictionary packages on the market from which to choose; most of these have similar

features. Therefore, the potential purchaser of a data dictionary is in need of guidance when making this choice. The United States Government has recognized this problem and has identified standards for data dictionaries in Federal Information Processing Standards promulgated by the National Bureau of Standards. An understanding of these standards and of the functions and objectives of a data dictionary will provide the reader with a basis on which to evaluate data dictionary packages and to use them effectively.

B. PURPOSE OF THE THESIS

We believe that it is important for managers in the military to understand what a data dictionary is and what it can do to help an organization manage its data. Thus, the purpose of this thesis is to provide the reader with an understanding of the structure and functions of a data dictionary, guidelines for the evaluation and selection of a data dictionary, and an analysis of several commercial data dictionary products. We will show the reader how the management of an organization's data resource can be accomplished by means of a data dictionary and will recommend ways for the role of the data dictionary to be expanded.

II. THE ANATOMY OF A DATA DICTIONARY

A. INTRODUCTION

Because data dictionary technology is a new and continually evolving field, it suffers from a lack of consistency in its terminology. The many texts and articles on the subject and the various commercial data dictionary products use a wide variety of differing terms. The data dictionary itself is known as a data dictionary/directory, a data dictionary system, or an information resource management dictionary. In order to provide a base of reference for the remainder of this thesis, we will present our own set of definitions distilled from our references.

Data dictionaries run the gamut from manual, on-paper systems to highly sophisticated software and can be used both in database and non-database environments. We will discuss automated data dictionaries only as they relate to a database, where they have the most to offer the potential user.

In order to assess the benefits of a data dictionary, it is necessary to understand how a data dictionary is organized and what its capabilities are. A data dictionary does not contain the actual data that constitutes an organization's database; instead, it is itself a database called a metadatabase that contains metadata, or data about the database data. Two types of metadata are found in a data dictionary. Dictionary metadata tells what data exists, the origins of the data, the attributes the data may have, how and by whom the data may be used, what the structure of the data is, and what the relationships between the data are. Directory metadata tells where the data is located, how it

can be accessed, and what its physical representation within the computer is. Together, these two types of metadata

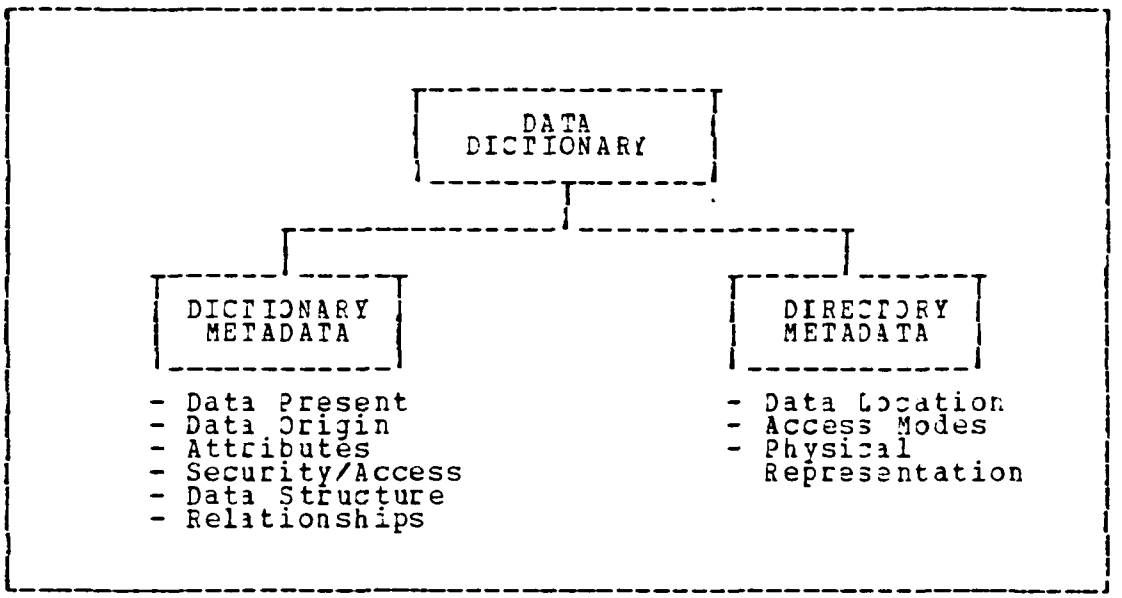


Figure 2.1 Types of Data Dictionary Metadata

provide the means for accessing and controlling the data in the database. Figure 2.1 illustrates this division of metadata.

Data dictionaries fall into two categories--free-standing and DBMS-dependent. Figure 2.2 shows a partial listing of some commercial data dictionary packages according to type. A free-standing data dictionary (also called independent or stand-alone) is not tied to any particular database management system (DBMS). It manages data by utilizing software routines built into the data dictionary package and thus is not dependent on DBMS software. This independence provides flexibility: a free-standing data dictionary can have the capability to support more than one type of DBMS. However, this flexibility is gained at the cost of duplication of data descriptions in the database and the data dictionary.

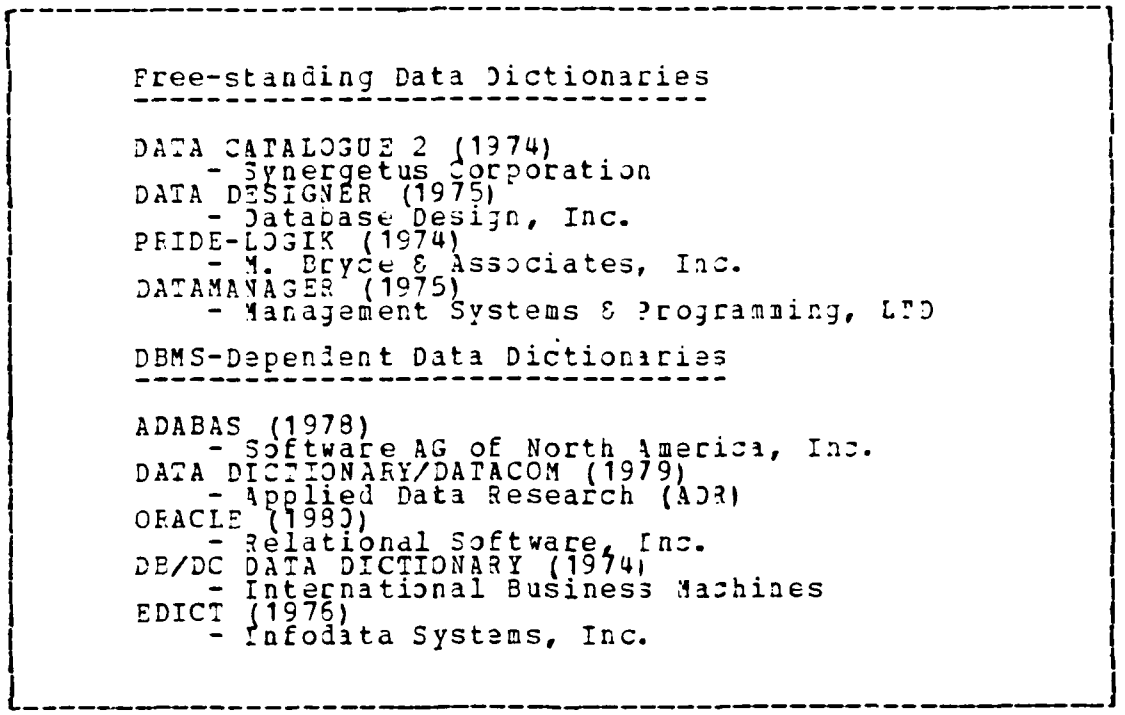


Figure 2.2 Free-standing and Dependent Data Dictionaries

A DBMS-dependent data dictionary (also called merged or integrated) is a component of a specific database management system; it uses the software facilities available within the DBMS to manage the data in the database. This type of data dictionary minimizes redundancy and limits the number of possible errors because data descriptions exist in only one place, in the data dictionary. It also benefits from the sophisticated backup and recovery facilities of the DBMS.

A data dictionary is also described as having active or passive interfaces or a combination of the two. An interface is a series of commands which connect the data dictionary with other software such as compilers, operating systems, report generators, and other programs. The data dictionary supports these applications by providing the metadata that is required for their execution. An active

data dictionary is one in which information is created, accessed, or modified through the data dictionary interfaces. New or changed metadata is automatically updated and stored in the data dictionary. This is not true of a passive data dictionary: when new metadata is generated, the data dictionary may or may not be automatically updated and when data is retrieved, it may be accessed through the data dictionary or directly from the database.

There are many perspectives from which to look at the data that resides in a database. There is the physical (or internal) view that consists of the actual physical representation, format, and location of the data as "seen" by the computer. There is a logical (or conceptual or global enterprise) view called a schema which describes all of the data in the database in its logical format, i.e., what types of records are to be maintained, the contents of those records, and the relationships among those records. This is the data as it would be presented to a human, not its actual computer format. In most cases, only the database administrator has access to the schema. Another view is the external view, also called a subschema, which is a subset of the logical view tailored to a particular user or application. This is analogous to a "window" through which only a portion of the total data is seen. Subschemas can be utilized to implement security by restricting a user's access to data.

Figure 2.3 shows the three different perspectives of data in a sample database of students at the Naval Postgraduate School. (A) is the computer's physical view and thus is not visible to the human user. (B) shows the overall logical view of this small database. (C) is a subset of (B) as it would be seen by a user who is interested in only a portion of the database--in this case, the senior Army officer who wants information only on Army students.

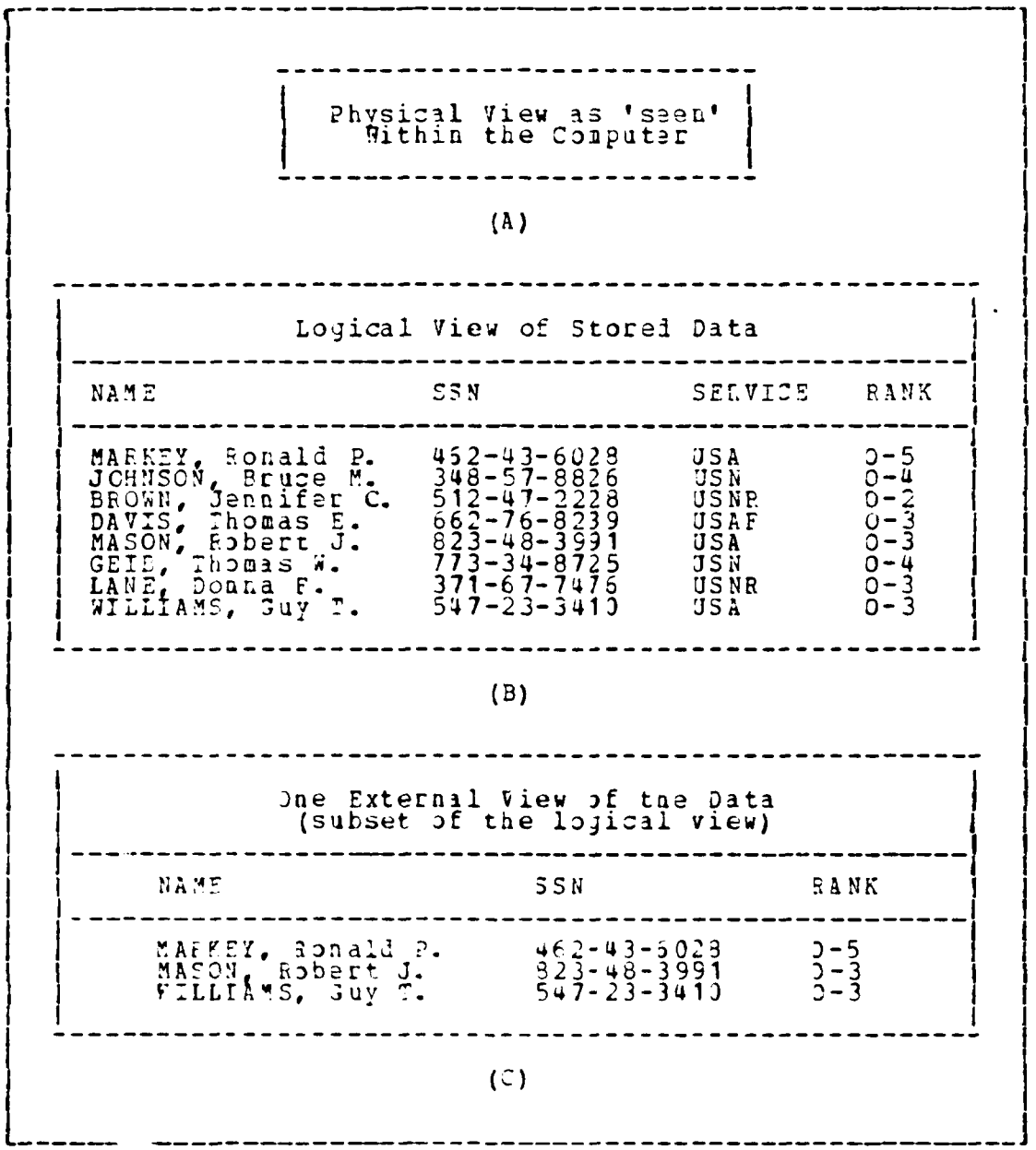


Figure 2.3 Views Within a DBMS

B. THE STRUCTURE OF A DATA DICTIONARY

There are three kinds of elements upon which the structure, or schema, of a data dictionary is built: entities,

attributes, and relationships. The basic element of the dictionary is the entity. Each entity has a unique name and represents an object in the real world, such as a person, thing, or idea about which information is recorded. For example, in our Naval Postgraduate School database we collected information about students. We also described the students by name, Social Security number, service, and rank. These characteristics of an entity are called attributes, and can be either quantitative or qualitative.

A relationship is a logical link between two entities that can also be described by attributes. A relationship will fall into one of three categories of mappings: one-to-one, one-to-many / many-to-one, or many-to-many. A one-to-one relationship exists when each entity or attribute is logically linked to one and only one other entity or attribute. For instance, we say that there is a one-to-one relationship between an individual's social security number and his name. In a one-to-many/many-to-one relationship, each entity or attribute is logically linked to one or more other entities or attributes. An example of this is the relationship between the instructor of a class and the students in that class. A many-to-many relationship occurs when one or more entities or attributes is related to one or more other entities or attributes. For example, there is a many-to-many relationship between the attributes "color" and "model" of a type of car--each color may be available on many different car models and each car model may be available in many different colors.

In order to understand the generic terms we have presented in their proper context, it is important to differentiate between the dictionary schema itself, the metadatabase that it governs, and the "real" data in the organization's database. These concepts are made even more confusing because the terminology used to refer to these

three levels of data differs from vendor to vendor and from author to author. We will look at these levels using the Applied Data Research, Inc. DATADICTIONARY terminology [Ref. 4] because it provides the clearest distinction between the three. (DATADICTIONARY will be discussed in depth in Chapter 7.)

At the highest level of abstraction, entities, attributes, and relationships are grouped by type:

> the dictionary schema can then be thought of as containing all existing entity-types, relationship-types, and attribute-types, any one of which will also be referred to as a schema descriptor [Ref. 5].

The schema descriptors are the general categories of data that is stored in the metadatabase. Figure 2.4 shows examples of some standard schema descriptors.

Entity-types	Attribute-types	Relationship-types
File	Author	Contains
Record	Description	Owns
Field	Password	Processes
Module	Status	Derived From
Program	Version	Resides
Report	Frequency	Uses
Job	Security Class	Includes
Dataview	Alias	Authority
User	Comment	Accesses
System	Effective Date	
Process	Usage Statistics	

Figure 2.4 Sample Schema Descriptors

At the metadatabase level, we look at specific instances of schema descriptors. Thus, we define an entity-occurrence as a specific instance of the general category entity-type. If PROGRAM is the entity-type, ACCOUNTS RECEIVABLE could be one entity-occurrence. Similarly, a relationship-occurrence

is a specific instance of the general category relationship-type. The relationship-type ACCESS may have as a relationship-occurrence PROGRAM-ACCESSES-FILE. At this level, we also talk about the specific characteristics of an attribute-type. An attribute-type is the name of a characteristic of an entity-occurrence, as Social Security Number characterizes a student. An attribute-characteristic is not the value of the attribute-type, but the parameters of an attribute-type, such as its length and format. For example, the attribute-type Social Security Number will be characterized as eleven digits long, of the form 999-99-9999. Entity-occurrences, relationship-occurrences, and attribute-characteristics will be referred to as the descriptors of the metadatabase.

At the "real" data level of the organization's database, we think in terms of actual values of data, such as "Jennifer C. Brown", "547-23-3410", "left-handed monkey wrench", "IBM 3033", or "93943". These are all values of the attributes of an entity, and are called attribute-values.

An example of each of the levels of data is given in Figure 2.5. We will use the generic terms entity, attribute, and relationship in this thesis where it is not necessary to distinguish between the three levels.

When a data dictionary is received from the vendor, it contains a system standard schema which includes certain basic entity-types, attribute-types, and relationship-types chosen by the vendor. A data dictionary is extensible if an organization is able to customize the schema by defining its own entity-types, attribute-types, and relationship-types in addition to those included in the system standard schema.

Schema Descriptors	Example
Entity-type	Record
Attribute-type	Name
Relationship-type	Contains
Metadatabase Descriptors	**Example**
Entity-occurrence	Student
Attribute-characteristic	25 characters, alpha-numeric
Relationship-occurrence	Student-Contains-Name
Database Data	**Example**
Attribute-value	Ronald P. Markey

Figure 2.5 Comparison of Data Levels

C. THE FUNCTIONS OF A DATA DICTIONARY

The functions performed by a typical data dictionary fall into four categories: definition, update, retrieval, and software interface. A data dictionary should be evaluated in each category according to the ease and success with which the functions are performed.

1. Definition

The first step in the implementation of a data dictionary is to collect information about some portion of an organization's data, such as the U.S.S. Constellation's supply department. This is done by interviewing supply department personnel, identifying the data received and produced by the department, and analyzing the software that manipulates that data. Once entities, attributes, and relationships have been defined, these data elements are entered into the data dictionary using the dictionary's data definition commands. The elements are classified according to the entity-types, attribute-types, and relationship-types of the

system standard schema, or the dictionary administrator may use customized data types as necessary, assuming the dictionary is extensible.

2. Update

As an organization evolves, so does its data. One of the functions of the data dictionary is to allow the addition, modification, and deletion of elements. For instance, a new Navy regulation might require the supply department to keep track of certain data about a new inventory item and to report this data quarterly. Or perhaps the administrative department will have to change zip codes to the new nine-digit format on all correspondence. Each of these changes will be introduced via modifications to the dictionary schema.

3. Retrieval

Information can be retrieved from a data dictionary by using query language commands or the report-generating capability of the dictionary. A dictionary will provide structured commands or an English-like query language that will help the supply department to find out the Navy part number for a monkey wrench. It will also allow the dictionary administrator to find out which users have access to a particular subschema. Reports are produced by a data dictionary according to a vendor-defined format or to user specifications. Reports generally produce a larger volume response than queries and are often printed out in hard copy.

4. Software Interface

The software interface function provides a means of access to the data dictionary for applications software, including compilers, editors, and database management

systems. A COPY command is used to bring data descriptions (e.g., of records or files) directly into the program being developed from the data dictionary. Thus, the job of the programmer is made easier and data use is standardized. It is also possible for applications software to directly retrieve and make changes to the elements in a data dictionary.

III. FEDERAL INFORMATION PROCESSING STANDARD FOR DATA DICTIONARY SYSTEMS

A. INTRODUCTION

The Institute for Computer Sciences and Technology of the National Bureau of Standards is in the process of developing a standard software specification for data dictionaries. The Federal Information Processing Standard for Data Dictionary Systems (FIPS DDS) is intended to serve as a guideline for the evaluation and selection of data dictionaries to be used by the federal government. The four volumes[1] "specify and describe the functionality, database structure, and user interfaces of the FIPS DDS" [Ref. 6].

We examined three volumes of the FIPS DDS: Command Language Interface Specifications (volume 2), Interactive Interface Descriptions (volume 3), and Dictionary Administrator Support Specifications (volume 4). The subject of each of the volumes corresponds to one of the three categories of users who will interact with a data dictionary--the experienced user, the relatively inexperienced user, and the administrator of the data dictionary.

The FIPS DDS describes in detail a suggested system standard schema for a data dictionary, including definitions and use of the schema descriptors. Each of the volumes presents the syntax for commands necessary for its target users to manipulate the dictionary. In addition, the results of each command are detailed, with error messages and "successful completion" messages listed where applicable.

---

[1]Note: Volume 1 is not yet available for review. The FIPS DDS is in draft form and has not been formally approved by the National Bureau of Standards.

B. SYSTEM STANDARD SCHEMA

The system standard schema set forth in the FIPS PUB provides basic entity-types, attribute-types and relationship-types as follows:

Entity-types

1. SYSTEM--a collection of processes and data
2. PROGRAM--an automated process
3. MODULE--an automated process which is a logical subdivision of a PROGRAM or an independent process called by a PROGRAM
4. FILE--an organization's data collection
5. RECORD--logically associated data which belongs to the organization
6. DOCUMENT--human-readable data collections
7. ELEMENT--data belonging to the organization
8. USER--members or collections of members belonging to the organization using the facilities available in the data dictionary
9. DICTIONARY-USER--users of the dictionary system itself
10. ACCESS-CONTROLLER--specifies access restrictions to an entity or set of entities in the dictionary

SYSTEM, PROGRAM, and MODULE are of the class "Process"; FILE, RECORD, DOCUMENT, and ELEMENT are of the class "Data"; USER is classed as "External", and DICTIONARY-USER and ACCESS-CONTROLLER are of the class "Security".

Attribute-types

There are 55 attribute-types included in the system standard schema, similar to the ones shown in Figure 2.4.

Relationship-types

The standard relationship-types provided by FIPS are as follows:

1. CONTAINS--describes entities composed conceptually of other entities
2. PROCESSES--shows the relationship between a process and data
3. RESPONSIBLE-FOR--shows the association between entities representing organizational components and entities denoting organizational responsibility
4. RUNS--shows the relationship between a user and a process
5. TO--shows the flow between two processes
6. DERIVED-FROM--shows that an entity is the result of some operation on another entity
7. The FIPS DDS includes an extensibility facility to provide for the customization of the system standard schema to match the organization's needs.

C. COMMAND LANGUAGE INTERFACE SPECIFICATIONS

The experienced user is one who is familiar with the structure and commands of the data dictionary and who needs access to the full functionality of the data dictionary. Command language commands are used to facilitate this access by allowing the user to:

--define data elements

--maintain the dictionary (add/modify/delete)

--report on dictionary elements

--query the dictionary about data elements

--build entity lists and perform operations on groupings of entities that meet certain criteria (useful for global, vice individual, operations)

--support applications programs that interact with the data dictionary

--perform general utilities, such as changing the mode of operation and obtaining help information.

The syntax of each of the command language commands is presented in the FIPS DDS using Backus-Naur form.[2] For example, the following command would be used to modify an entity that already exists in the dictionary:

```
MODIFY-ENTITY
  {[WHERE] NAME [IS] <entity-name>
    [ADD NEW-VERSION [<version-number>]]
    WHERE ATTRIBUTES [ARE] <attribute-clause-1>
     [,....., <attribute-clause-n>]}
```

where:

--entity-name refers to a single entity in the dictionary

--NEW-VERSION is an optional clause which results in the creation of a new entity which has a primary-name consisting of the assigned-name of the entity-name specified and the next-highest version-number

--attribute-clause-n refers to a clause used to designate the attributes of the specified entity which are to be modified

D. INTERACTIVE INTERFACE SPECIFICATIONS

The interactive interface for the relatively inexperienced user is designed to lead the user step-by-step through the desired operations. Without having to master the command language commands, the interactive interface user has a large subset of the total functionality available within the data dictionary, including manipulation, reporting, querying, and entity list operations. The FIPS DDS recommends that this interface be implemented by means of "panels" (screens) that are presented to the user in sequence and which contain the following information areas:

---

[2]Backus-Naur form is explained in Appendix A.

1. state area--tells the user where (in which dictionary) he is and what he is doing
2. data area--for entering and displaying data
3. schema area--used mostly for dictionary updates to show available options and limitations on actions
4. message area--for error messages and warnings
5. action area--tells the user how to proceed from the current panel
6. help area--for the display of help information requested by the user

The user begins his session with the data dictionary at a "home panel" which provides entry into the system. At any point along the way he has the option of saving or undoing any panel with which he has been working. This panel-driven interface ensures that the user always knows where he is in the dictionary, what mistakes he has made, what choices he has to continue, and what help is available to him.

E. DICTIONARY ADMINISTRATOR SUPPORT SPECIFICATIONS

The administrator of the data dictionary, of course, has access to both the standard command language and the interactive interface. His or her main concern, however, is the management of the schema. This is accomplished by means of a specialized set of commands for

--extending the system standard schema

--reporting on the schema

--implementing access control measures

--controlling export from and import to the dictionary.

We have already defined the extensibility facility as the ability to add schema descriptors to the system standard schema. The report facility allows the administrator to generate a listing of the entire schema or any subset thereof. The security facility provides commands for

restricting the access of users to the dictionary by specifying which commands the user is allowed to execute. The export/import facility allows transfer of parts of one dictionary to another, but only between dictionaries whose schema are identical in order to preserve the integrity of the "target" dictionary.

F. EVALUATION

It is certainly true that the FIPS DDS presents the reader with very detailed specifications of the commands and facilities for a standardized data dictionary; the volumes we reviewed could serve as the basis for an initial design specification for the development of data dictionary software. A dictionary based on the FIPS specifications would perform the required functions discussed in Chapter II and would contribute to the organization's management of its data. The military and the federal government would benefit greatly from the availability of standard software to achieve control over its data resource.

The major contribution of the FIPS DDS is its orientation to the needs of the different kinds of users of a data dictionary. This is particularly evident in the interface that is suggested for use by inexperienced users of the data dictionary. The panel-driven format with its six information areas is far less intimidating than the syntax required by the command language. Even so, the interactive interface still requires a certain degree of sophistication on the part of the "inexperienced" user if he is to be able to manipulate the dictionary. Another strong point of the FIPS DDS is its consistency of presentation and format. No matter what the operation, the procedures needed to manipulate the dictionary and the manner in which the dictionary "responds" to the user are logical and predictable. The

commands, however, are complex and require knowledge of Backus-Naur form.

Even though the FIPS DDS does indeed provide a comprehensive software standard for the computer professional, we do not believe that it achieves its goal of providing a guide for the evaluation and selection of data dictionaries. Although the addition of the introductory volume may help remedy the problem, the three volumes of specifications ignore the forest of reasons behind the implementation of a data dictionary while concentrating solely on the patterns of the leaves on each tree. The FIPS DDS will not be extremely useful to the individual searching for basic assistance in evaluating commercial data dictionary packages. Many of the books and articles we have reviewed provide better explanations of data dictionary features and comprehensive evaluation criteria.

We found that the terminology that the FIPS DDS uses for the dictionary schema and the metadatabase is not explained clearly nor is it any less confusing than that of any other publication. In addition, no specific examples of how an organization's data would be entered in the data dictionary are given. We feel that it is more important for the potential data dictionary user to understand how a data dictionary will assist in the management of data than to see samples of every conceivable type of error message that could occur. A summary of recommended features such as the one we have just presented and a list of criteria for evaluation would be far easier for the reader to digest.

None of the data dictionary packages we have reviewed does things totally the "FIPS way", and it is unlikely that any commercial dictionary vendor will ever conform exactly to FIPS DDS guidelines. However, it is likely that the federal government will insist that FIPS standards be incorporated into future dictionaries intended for govern-

ment use. In the next chapter we will develop a set of criteria for an "ideal" data dictionary, taking FIPS DDS recommendations into account. In Chapter V we will examine four commercial data dictionary packages and evaluate their success in meeting the ideal criteria.

IV. THE ROLE OF THE DATA DICTIONARY IN INFORMATION RESOURCE MANAGEMENT

In this chapter we will see how a data dictionary can contribute to the goal of efficient management of an organization's data. We will first discuss the process of development of an information system in an organization and then will discuss the three objectives of data dictionaries that we have identified as contributing the most to the accomplishment of this goal: data security, data integrity, and documentation/maintenance. We will then develop a set of criteria for the "ideal" data dictionary to be used in the evaluation of data dictionary packages.

A. INFORMATION RESOURCE MANAGEMENT

Organizations today have become increasingly aware of the need to manage data just as they manage other essential resources. If properly managed, the necessary data will be available, up-to-date, and retrievable when required to provide information that is of value to the organization. This concept is known as Information Resource Management, or IRM, although it might also be referred to as Data Resource Management.

IRM has been the focus of a great deal of interest in recent years. In October of 1980, the Institute for Computer Sciences and Technology of the National Bureau of Standards (NBS) and the Association for Computing Machinery (ACM) co-sponsored a workshop on IRM strategies and tools. It was based on the premise that

> IRM is currently one of the most significant topics being discussed concerning information systems, and is being discussed along a variety of lines of thought.

> These include business systems planning; information systems analysis, design, and development; database design and implementation; the disciplines of office management, paperwork management, and information sciences management; and the various problems and costs associated with implementing IRM to include each of these areas. [Ref. 7]

The Proceedings of the workshop defined IRM as

> whatever policy, action, or procedure concerning information (both automated and non-automated supported) which management establishes to serve the overall current and future needs of the enterprise. Such policies, etc., would include considerations of availability, timeliness, accuracy, integrity, privacy, security, auditability, ownership, use, and cost effectiveness. [Ref. 8]

The recommendations of the NBS/ACM workshop on the role that the data dictionary should play in IRM were incorporated into the Federal Information Processing Standard for Data Dictionary Systems that we discussed in Chapter III.

In order to understand how the data dictionary contributes to the production of valuable information for an organization, we will look more closely at the organization itself and at its functions. An organization is made up of many systems that convert resources into usable output. An information system, then, is one that takes raw data and transforms it into information that can be used by the organization. If the process by which the organization develops its information systems is the heart of information resource management, then it is the data dictionary that keeps it ticking.

Assume that the U.S.S. Constellation has identified a problem with the way a particular information system is currently operating--it could be preventive maintenance record-keeping, the supply department inventory, the personnel administration system, or a system that affects the entire organization. The process of analyzing the

system and developing a system to solve this problem evolves through four distinct phases, called the System Development Life Cycle (SDLC). We will show how the data dictionary supports the SDLC, and thus, IRM, through planning, study, design/coding, and operation and maintenance. We have based our analysis of the SDLC on that of Leong-Hong and Plagman [Ref. 9].

1. Planning Phase

The Proceedings of the NBS/ACM workshop emphasized the need for a "top-down" approach to IRM in an organization. During the planning phase, the organization's long-range plans, its functions, and structure are analyzed to ensure that any information system that is developed will complement those needs.

If a data dictionary is already in existence, it can provide information about the functions of the organization that have been defined, or it can document the initial definition of those functions. For each function, it must be determined who does it, what is produced, what other functions it interacts with, and what inputs are needed to accomplish the function. As an example, we can say of the Payroll function that it is performed by the disbursing office, paychecks and leave and earnings statements are produced, it interacts with the personnel administration system, and it requires data about all members of the crew, including rank/rate, time in service, and so on.

At this stage of the development process, the "big picture" is drawn while the details are left until later. Thus, general categories of data such as "accounting data" and "personnel data" and the transactions that affect them are defined and entered in the dictionary.

In the aggregate, this planning information constitutes a conceptual data model. "Definition and analysis of

subsequent information requirements (and eventually, database design) will be dependent upon this data model" [Ref. 10]. The fact that the development of this model has been automated, rather than manual, ensures a quicker, standardized process.

2. Study Phase

At this point in the SDLC, a greater level of detail is introduced. The data dictionary provides a common, standardized source of information about the inputs and outputs of the organization's functions. Specific entities, attributes, and relationships are chosen from the general categories of data identified in the planning phase. The entity PART in the Constellation's inventory system may be described by the attributes Navy Part Number, Description, Storage Location, and Quantity. There may also be a many-to-many relationship assigned between PART and DEPARTMENT. Reports required to be produced are also defined and the necessary input data is identified.

This information provides what is called a detailed conceptual model, an expansion of the conceptual model of the planning phase. The data dictionary can be used to identify redundancy within the data model by determining whether the data entered already exists. In addition, with the aid of the dictionary, the systems analyst will be

> able to determine what data is available, how it is being used, how it can be accessed, who has primary responsibility for its definition and upkeep, and most important, whether there is conflict in using this data, that is, what impact it will have on other application systems [Ref. 11].

3. Design/Coding Phase

The purpose of the design phase is to provide specifications for programming and implementing the system. It is here that the data dictionary's schema descriptors will be used or expanded to meet the needs of the system. If a database does not already exist, and it is determined that one is required, the data dictionary schema will provide a basis from which to implement one. Data integrity is enforced because the dictionary serves as the sole source of data definition and structure.

When software is being coded, the data dictionary provides documentation for the programmer and a COPY facility for transporting record definitions, for example, into the program being developed. An important element of the dictionary is the constraints that are defined for data values. In this way, data that is input to a program can be checked against the constraints that have been established. Documentation of the program includes the author, a description, input requirements, output produced, and information on what other programs are called upon, all of which are incorporated into the data dictionary.

4. Operation and Maintenance

After a new system has been implemented, the work of the data dictionary does not end. All of the documentation that has been recorded during the development of the system serves as a base of reference for the users of the system. In addition to the database administrator and the administrator of the dictionary, the key players in information resource management who benefit from the use of a data dictionary fall into six groups, according to Allen, Loomis, and Mannino: [Ref. 12]

1. The data administrator, who is responsible for the overall administration of the data resource, uses the dictionary as a tool to enforce the way data is stored, maintained, and monitored.
2. Data processing managers benefit from the dictionary's reports on data usage.
3. Operations personnel retrieve information from the dictionary about jobs that are being run.
4. Programmers and analysts use the dictionary to retrieve data definitions and to document a system being developed.
5. End users access the data dictionary for descriptions of their dataviews.
6. Finally, auditors will use the documentation provided by the data dictionary to trace data and programs as they are used in the computer system.

It is the process of implementing a data dictionary that we have just described--the analysis of the organization, the definition of its functions, and the documentation of its information systems--that makes the dictionary so important in information resource management. We have seen that during the development of an information system, the data dictionary is involved from the initial planning stage, through the programming process, through the operation, and into the maintenance of the system. The dictionary provides the standards for data which will be used throughout the life of the system and referenced when developing other systems. Key contributions include decreasing the amount of redundancy of data required to be stored, enforcing security of the valuable data resource through access controls and implementation of user views, and providing documentation which serves as a "corporate history" and as a reference upon which maintenance and auditing are based. These objectives of data dictionary usage are discussed in detail in the next section.

B. OBJECTIVES OF A DATA DICTIONARY

In this section we will focus on the three major contributions of the data dictionary to the management of an organization's data. These are data security, data integrity, and documentation/accountability. Although we recognize that other objectives of data dictionary usage might be identified, we believe that each will fall into one of these three major groupings.

1. Data Security

There are two distinct levels of security of the data in an organization's database which will be provided either by the data dictionary or by the database management system itself. First, procedures should exist to ensure that only authorized personnel are allowed to access the information contained within the database. The widespread use of computers and the increasing sophistication of users has made an organization's data vulnerable to embezzlers, amateur "hackers", corporate spies, and careless employees. Second, the system should contain provisions for controlling the amount and types of data that each authorized user is allowed to access within the system. Some of the sophisticated data dictionaries, for example, include a trace mechanism which increases security by recording every inquiry that is made into system files and data. If an intrusion is made into the system by unauthorized personnel, the specifics of that inquiry, including the data which was accessed, will be recorded.

Metadata should be afforded at least the same protection, if not more, than the data in the database. Leong-Hong and Plagman [Ref. 13] present an example of the importance of the security of metadata as it concerns

the data resources in intelligence/military applications such as the classification code of intelligence documents. When security profiles for the metadata entities are stored in the metadatabase, unauthorized access to the metadata could be most damaging. This is because presumably one would be able to 'crash into' the system using that information.

It is the task of the dictionary administrator to analyze the metadata to determine the levels of security required and to grant access privileges (read and write, read only, update) to users for certain portions of the metadata. Information about users, their password, and privileges is stored in the data dictionary and is accessible only to personnel authorized by the administrator.

We have already shown in Figure 2.3 that subschemas contribute to security by limiting the size of the "window" through which a database user looks at data. When a user attempts to access a particular subschema, the request is routed through the data dictionary to determine whether access is authorized and, if so, the structure of the subschema. Only at this point is the "real" data in the database accessed.

2. Data Integrity

The keys to data integrity are the control of inputs to the database and the minimization of data duplication. Properly used, these keys will enhance communication between users by ensuring that a single, correct source of data is maintained.

Because the data in a database is shared among many users, it is essential to have some means of enforcing standards for entering data, updating it, and maintaining it. For example, the data dictionary identifies constraints, or limitations on the values data can have. Fields can be defined as being mandatory or optional, alphanumeric or numeric, and a minimum or maximum length. The data

dictionary contains comments on how data should be used in order to assist those using the data dictionary. Another important control feature of a data dictionary is how it deals with synonyms--an entity or attribute with more than one name. For instance, the entities EMPLOYEE, REGIONAL_MANAGER, and EXECUTIVE may all be used by different departments in the organization to refer to Linda Smith. The administrator must standardize the terminology used in the organization and eliminate as many synonyms as possible. When this is not feasible, all of these synonyms, or aliases, must be recorded in the data dictionary. Van Duyn [Ref. 14] explains that

> It is not unusual to have similar types of data elements in the database and in various applications. In such cases, and in cases where the same data type is known by other names, the DDS [data dictionary] can be used to inform the users of the relationships that exist among these data and of the disposition of their usage. In other words, the DDS provides information as to which modules/programs and systems use the same data type and how they relate.

The data dictionary also contributes to data integrity because it reduces the necessity for duplication of data and therefore lessens the opportunities for error. The information about the components of different subschemas of the same logical view is stored in the data dictionary in place of the data itself. A user, whether writing a program or creating a new entity-type, should be able to query the data dictionary to ensure that the necessary routines or entities do not already exist within the system. Perhaps

> one of the most important benefits of DDS [data dictionaries] is that because it gives accurate and timely information, management can control more efficiently not only the automated and manual data of the enterprise but all its resources and operations. Consequently, management is provided with precise and accurate data for quick, profitable decision-making [Ref. 15].

Thus, the possibility of two users querying the database and receiving different answers to the same question at the same time is decreased.

3. Documentation/Maintenance

Because maintenance is the most expensive and time-consuming phase of software development, documentation and maintenance of the organization's data is probably the most significant objective of the data dictionary. It is a fact of software life that documentation is often avoided during system development and program design. To a large extent, this is because documentation can be prepared as an "afterthought"; it is not essential to the operation of the system. But when a system is developed that includes a data dictionary from the beginning, the data which is required by the data dictionary forces documentation to become an integral part of the design. "The use of a dictionary provides documentation of a quality and form that is simply not available through less formalized procedures in the data processing environment" [Ref. 16].

The data dictionary can also reduce the amount of effort required by maintenance personnel because it provides "a 'roadmap' for the programmer doing maintenance. It records the programs being maintained, their data structures and their relationships" [Ref. 17]. We have defined an active data dictionary as one in which information is created, accessed, or modified through the data dictionary interfaces with new or changed metadata automatically stored in the data dictionary. This "continuous maintenance" can be used to allow the database administrator to monitor where data is used, who uses it, how often it is used, and what changes have been made to it. Because the data dictionary provides a wealth of documentation, it is possible to trace an "audit trail" through the organization's data, from user

names and department to the kind of data used in a program to how many records a certain field appears in. Also,

> The tracking of how programs/modules use particular data as well as which files/segments contain certain data is extremely important to the systems analyst in performing system changes. Through the DDS [data dictionary], he or she is able to ascertain what impact the proposed changes will have on other components of the system and upon functional areas within the enterprise. By having an accurate, up-to-date assessment of the location and usage of data that will be involved in the system change, the analyst can accomplish the task more efficiently [Ref. 18].

Once an organization has decided to make a commitment to manage its data using a data dictionary, it must decide what kind of data dictionary best suits its particular needs. In the next section, we will look at the features of what we have called the "ideal data dictionary" as a basis for evaluating the many commercially available data dictionary packages from which the organization must choose.

C. THE IDEAL DATA DICTIONARY

Having identified the functions of a data dictionary in Chapter II and how they support the accomplishment of the objectives just discussed, it will be helpful to use these concepts to evaluate data dictionaries. The "ideal" data dictionary would be one that possesses all the capabilities necessary to support all potential users in all possible applications. However, this ideal dictionary would be impossible to conceptualize, much less to create. The ideal data dictionary for an organization will depend on the organization's size, functions, and needs. The potential users of a dictionary will have to develop a set of criteria upon which a candidate will be judged.

Many references provide criteria for evaluating data dictionaries and identify those characteristics which are vital to the management of the data resource. Unfortunately, it is difficult to find two references that propose the same criteria. One excellent source, Leong-Hong and Plagman [Ref. 19], lists nine categories for evaluation:

1. data description facility
2. data documentation support
3. metadata generation
4. security support
5. integrity support
6. user interface
7. ease of use
8. resource utilization
9. vendor support

It is important to recognize a distinction between two categories of criteria for the ideal data dictionary: those that evaluate the vendor and operating environment, and those that evaluate the data dictionary itself. In the former category, items like vendor support and reliability, the choice between free-standing or DBMS-dependent data dictionaries, the degree of integration with other system components, and the quality of system documentation are important considerations that may drive the decision between two comparable data dictionaries. It is, however, the latter type of criterion that will be vital in identification of the essential requirements of the ideal data dictionary. We have grouped all such requirements into six categories: system standard schema and extensibility, command and query languages, ease of use (including menus), security, documentation and reports, and application interfaces. (We have assumed that the objective of data integrity will be accomplished by the correct, and enforced, use of any data dictionary.) If a particular dictionary fully

supports each of these six criteria then it will most likely meet all of the organization's data management needs.

1. System Standard Schema and Extensibility

The ideal data dictionary must provide a system standard schema with all the descriptors necessary to support the range of applications required by the organization while still being simple enough to be competitively priced. It must provide "enough" descriptors to be fully capable without providing so many that the schema becomes confusing. Additionally, the ideal dictionary must support the user (or data dictionary administrator) in modifying existing schema descriptors and creating new entities, relationships, and attributes. This extensibility is vital in supporting applications specific to the organization's needs.

2. Command and Query Languages

The ideal dictionary must provide both command and query languages. The command language must support creation and modification of data structures and subsequent entry of data into those structures. The command language must include edit commands to facilitate addition, modification, and deletion of system data. It should include commands restricted to use by the data dictionary administrator, e.g., password assignment. The ideal system will include a query language to support the analysis and production of usable information from the organization's data. Perhaps one of the most important features of a data dictionary (and database), query languages allow data to be screened in order to provide concise and specific information to support timely management decisions.

3. Ease of Use

Ease of use, or user-friendliness, is another important aspect of the ideal data dictionary. It must be supportive of new users while still providing full functional support of the system "experts". Two primary ingredients of user-friendliness are the availability of menus and carefully conceived examples in the dictionary's reference manuals. A hierarchy of menus can reduce complex operations to a series of smaller, friendlier steps while user documentation provides easy-to-understand examples that guide the inexperienced user through each phase of system operation. As microcomputers and the concept of the automated office continue to spread, ease of use will become an even more important consideration in deciding which software products to utilize.

4. Security

Security will be a vital concern of the ideal data dictionary. Protection and control of system information must be provided. The data dictionary administrator must be provided the capability to control personnel access to system data. He or she must also be able to grant different degrees of access to different users. Similarly, users should have the capabilities to protect, and grant access to, those structures and data which they control.

5. Documentation and Reports

The documentation and reports created by the ideal data dictionary must also be clear and understandable. Timely and accurate preparation of reports is a key objective of any DBMS. The data dictionary is uniquely qualified to assist with this function. By ensuring the integrity of data accessed and supporting query commands, the ideal data

dictionary can provide reports and documentation to answer specific questions as they arise.

6. Application Interfaces

The final important characteristic of the ideal data dictionary is its ability to interface with the other applications that may exist in the organization. If the data dictionary is free-standing, it should interface with many of the currently available database management systems. If DBMS-dependent, the dictionary should interface with all components of that system. Additionally, the ideal data dictionary should interface with code generators, communication systems, and other agents of the users' environment.

In the following chapter, we will study and evaluate four of the popular data dictionaries that are currently available. We will use these characteristics of the ideal data dictionary that we have defined to compare and contrast the features of the four dictionaries. In addition, each will be compared to "standard" dictionary presented in the FIPS DDS.

V. EVALUATION OF COMMERCIAL DATA DICTIONARIES

The purpose of this chapter is to review and evaluate a cross-section of commercial data dictionary packages. We selected four dictionaries: DATA DESIGNER, DATAMANAGER, ORACLE, and DATADICTIONARY. User documentation and library sources were the primary sources of information for our evaluation. Additionally, ORACLE was available on the Naval Postgraduate School's Vax minicomputer, and we observed demonstrations of DATA DESIGNER and DATADICTIONARY.

A. DATA DESIGNER

DATA DESIGNER is a free-standing data dictionary developed by Database Design, Inc. It was introduced in 1975 with the goal of supporting logical database design by solving some of the traditional problems associated with muliple-application database management systems, such as duplication of data, excessive storage requirements, data consistency, complexity, and modifiability. DATA DESIGNER can be used in conjunction with a variety of database management systems, including IMS, IDMS, ADABAS, NOMAD, and others. Additionally, it can produce designs that will interface with COBOL and other non-DBMS tools or systems.

DATA DESIGNER can be characterized as an

> automated, easy-to-use tool that assists the database designer in formulating normalized views of the data requirements and synthesizes these views into a canonical normalized form. . . . DATA DESIGNER maintains information needed to physically structure the database for efficient performance [Ref. 20].

In addition to providing the standard functions of a data dictionary, DATA DESIGNER goes several steps beyond. It

provides an extensive set of commands categorized as user commands, edit commands, and plotting commands, as shown in Table 1. It also supports limited production of models and graphics. Furthermore, DATA DESIGNER's capabilities include powerful generation options and report features that will support the design and maintenance of applications.

TABLE 1

Standard Commands of DATA DESIGNER

User Commands

ADD	BATCH	BUILD
COPY	CREATE	EMPTY
END	FILES	GENERATE
HELP	HIERARCHY	PLOT
PRINT	RENAME	REPORT
SHOW OPTIONS	TRANSFER	VALIDATE

Edit Commands

DELETE	EDIT	INSERT
LIST	RENUMBER	REPLACE

Plotting Commands

DRAW	DONE	RETURN
SET ALL	SET DEVICE	SET RANGE
SET TITLE	SET TYPE	SHOW

DATA DESIGNER supports logical database design through a five-step process:

1. A data dictionary file is created that contains a list of all standard data item names to be used.
2. Subschema files are created that describe all of the views necessary to support user data requirements.

3. The encoded user views are validated. This step verifies the syntax of each view and ensures that each data item name listed in a view is in the data dictionary.
4. All of the verified user views are synthesized into a logical data model. Reports and diagrams are generated to reflect this model.
5. The model is evaluated to ensure that it meets all user requirements and is modified as necessary by repeating steps (1) through (4).

DATA DESIGNER utilizes three kinds of files: dictionary files, subschema files, and generated design files. A dictionary file ($DIC) contains a list of all data elements that will be used in an application or subschema. This list serves as a base for further development, e.g., additional views. A subschema file ($SUB) contains data items and relationships pertaining to particular views. Finally, the generated design file ($DES) contains a logical data model generated by DATA DESIGNER using the applicable dictionary and subschema files as input. The generated design files, in turn, serve as the input for the report and graphics functions.

Key commands utilized during the creation of a logical database design include the following:

CREATE--defines dictionary and subschema files.

BUILD--enters data item names into created files.

VALIDATE--compares the subschema files to the dictionary file.

GENERATE--creates a logical DB design from the validated files.

REPORT--prepares design documentation for the logical design.

PLOT--uses the plotting subsystem to draw the logical design.

EDIT--supports modification of existing files when necessary.

In order to acquaint the reader with the operation of DATA DESIGNER, we will demonstrate the dialog associated with each step of the process necessary to create our Naval Postgraduate School database example of Chapter II. The user of DATA DESIGNER must first create the dictionary file STUDENT.DIC and the subschema file STUDENT.SUB (user inputs are indicated by boldface type) as follows:

```
 >CREATE STUDENT.DIC DICTIONARY
DDFC0101I File "STUDENT.DIC" of type "$DIC" created.
 >CREATE STUDENT.SUB SUBSCHEMA
DDFC0203I File "STUDENT.SUB" of type "$SUB" created.
```

Next, the BUILD command is used to load data items into the two created files. First all possible data items are listed in the dictionary file:

```
 >BUILD STUDENT.DIC
DDBS0065I The file type is $DIC.
DDBS0018I There are no records in the file.
 B>NAME
 B>SSN
 B>SERVICE
 B>RANK
 B>DONE
DDBS0064I File building is done.
DDBS0068I 4 records were entered
DDRN0098I Line 1100 is now the last line in your file.
```

The subschema file will support creation of one or more user views. In our example, the subschema file contains two views, the basic, overall view and the view intended for Army use only. Notice that after the user enters the BUILD process, each line must start with a modeling code. These codes are used to identify components and to establish relationships within the views. When building the subschema files, all desired relationships must be specifically stated. DATA DESIGNER uses "1" to specify a one-to-one relationship and "M" for a one-to-many relationship. A complete list of the modeling codes used in this example appears in Table 2.

```
 >BUILD STUDENT.SUB
DDBS0075I The file type is $SUB.
DDBS0081I There are no records in the file.
 B>V,STU-1
 ****************************************
 *  THIS VIEW SUPPORTS THE OVERALL VIEW  *
 ****************************************
 B>F,0100
 B>T,0003
 B>K,SSN
 B>1,NAME
 B>1,SERVICE
 B>1,RANK
 B>V,STU-ARMY
 ****************************************
 *  THIS VIEW SUPPORTS THE ARMY VERSION  *
 ****************************************
 B>F,0125
 B>T,0002
 B>K,SSN
 B>1,NAME
 B>1,RANK
 ****************************************
 B>DONE
DDBS0064I File building is done.
DDBS0068I 13 records were entered
```

TABLE 2

DATA DESIGNER Modeling Codes

Code	Modeling Use
V	Name a user view
F	Specify frequency of use
T	Specify req'd response time
K	Name a key
C	Concatenate keys and data
S	Concatenate keys in short way
L	Label a data group
M	Identify a multiple association
1	Identify a single association
N	Name an association
*	Insert comments

Once the dictionary and subschema files are formatted, the VALIDATE command is used to ensure that all entries and relationships in the subschema files are valid based on the information previously specified in the dictionary file.

DATA DESIGNER will respond with the number of views processed, the number of lines read, and the number of validation errors, if any, that were located:

```
>VALIDATE STUDENT.SUB STUDENT.DIC

DDVS0013I Validation begins.
DDVS0024I 2 Views were processed.
DDVS0025I 13 lines were read.
DDVS0015I 0 validation errors were detected.
```

Once the files are successfully validated, the user will utilize the subschemas to generate a logical database design for his or her application.

The ten GENERATE options from which the user can choose are powerful features that allow the user to control the way that DATA DESIGNER produces a design and supports requests for varying degrees of information during the generation process. If the GENERATE command is called without options, DATA DESIGNER will create a design that removes all redundant data elements, generates intersection data groups as necessary to resolve many-to-many relationships, suppresses repeating data elements within data groups, generates single key data groups from concatenated keys, and considers all frequency and timing information that was contained in the subschema files. In all cases, the end product of the GENERATE command will be creation of a $DES file, in this case, STUDENT.DES. The factory user's guide recommends that options 4, 5, 6, 9, and 10 be used when generating the initial design or after major revisions to the input files. A brief description of each generate option is shown in Table 3. Continuing with our student database example, the user's dialog will be

```
>GENERATE OPTION 4 5 6 9 10 TO STUDENT.DESIGN
DDGS0032I Design generation begins.
DDGS0058I The subschema file is STUDENT.SUB
DDGS0214I Option 4 ignores undefined links.
DDGS0281I Option 5 generates foreign key information.
DDGS0301I Option 6 generates candidate key information.
DDGS0307I Option 9 generates cross-reference info.
DDGS0307I Option 10 ignores frequency and timing info.
DDGS0054I Design generation has finished.
```

TABLE 3

DATA DESIGNER Generate Options

Option	Purpose
1	Generate unspecified associations.
2	Suppress resolving redundant data.
3	Suppress creating intersection files.
4	Supress generating inverse links.
5	Generate foreign key information.
6	Generate candidate key information.
7	Allow repeating data items in groups.
8	Suppress generating single key groups.
9	Generates cross-reference information.
10	Suppress frequency/timing information.

At this point, the logical database design is completed. When using the options specified in the example, a series of reports will be automatically generated. A list of reports

TABLE 4

Reports Available with DATA DESIGNER

Report	Type
1	Data Group Links Report
2	Canonical Schema Report
3	Data Group Index Report
4	Multiple Occurences of Data Items.
5	Data Relation Report
6	Data Group Candiates Keys Report
7	Data Item to User View Cross-Reference
8	User View to Data Group Cross-Reference
9	Data Group to User View Cross-Reference

created is contained in Table 4. To print these reports, the user's dialog will simply be

```
>REPORT 1 2 3 4 5 6 7 8 9 PRINTER FROM STUDENT.DESIGN
DDP20073I The reports were printed.
```

As a final aid in evaluation of the logical database design, DATA DESIGNER is capable of producing diagrams of (1) an overview of the logical database design and/or (2) a hierarchical representation of that logical design. To produce the logical overview diagram, the following dialog is required:

```
>PLOT
DDPT0289I       DATA DESIGNER Print Plot Release 2.5A
 P>SET TYPE OVERVIEW
 P>SET TITLE LOGICAL-DESIGN
 P>DRAW FROM STUDENT.DESIGN
DDFS0310I Design STUDENT.DESIGN's description loaded.
DDNX0271I The overview plot generation is done.
 P>RETURN
 P>END
```

After using the printed reports and diagrams to evaluate the database design, the user will, if satisfied, transcribe the design into a specific DBMS format, such as ADABAS, or use DATA DESIGNER's EDIT capabilities to revise the design as necessary.

As discussed in Chapter IV, data dictionaries can be evaluated on the basis of their accomplishment of security, integrity, and documentation/maintenance. DATA DESIGNER, as a free-standing data dictionary that can be used in conjunction with a variety of DBMS and non-DBMS systems, does not address the security aspect. It was apparently designed with the assumption that the parent system with which DATA DESIGNER interacts will handle access control and other security-related functions.

DATA DESIGNER does, however, receive high marks for maintaining data integrity and for the quality of its documentation. Because it is designed to support the development of logical database designs, it utilizes its dictionary files to ensure that duplication of data is prevented through generation of cross-reference files. When a subschema is modified, DATA DESIGNER again utilizes its

dictionary files in the subsequent design generations. The PLOT and REPORT functions provide a wealth of information about the design, its components, and all users of the subschemas. Relationships, both those included by the user and those produced by DATA DESIGNER, can be seen in written reports and visual representatinons. When modificatons and new designs are produced, the reports are automatically uplated to reflect all changes.

B. MSP DATAMANAGER

DATAMANAGER, developed by MSP, INC. of Lexington, Massachusetts, is one member of the MANAGER family of dictionary-oriented software products. Other products include DESIGNMANAGER, PROJECTMANAGER, SOURCEMANAGER, and TESTMANAGER. The entire line of products, while capable of batch operations, is designed specifically to support interactive operations with IBM 360/370/30xx/4300 series (and plug compatible) computers. While DATAMANAGER is designed as a nucleus for further expansion or specialization, it provides all basic capabilites necessary to create and maintain user dictionaries. Additional capabilites, available as a series of extra-cost, add-on modules, include:

1. interfaces to IDMS, ADABAS, IMS, TOTAL, SYSTEM 2000, and other DBMS
2. teleprocessing interfaces
3. generation of COBOL, PL/I, or other source language data descriptions
4. generation of DATAMANAGER data definitions from existing COBOL or PL/I source code
5. interfacing of a DATAMANAGER dictionary to user-written programs
6. status, audit, and security facilities
7. extensibility through a user-defined syntax facility

DATAMANAGER can provide data dictionary capabilities to users utilizing a variety of hardware/software combinations. By providing interface modules for several popular database management systems, DATAMANAGER is obviously more flexible than one that is tied to a single, distinct database system. However, DATAMANAGER's flexibility extends beyond the obvious:

> DATAMANAGER is intended for use in any organization in which there is a computerized data processing function. Its use, however, is not confined to those elements of data that are held in computer files or that are acted upon by computerized systems. Definitions of all data held and used by an organization, in its manual systems as well as its computerized ones, can be held in a DATAMANAGER data dictionary. DATAMANAGER is designed to be used both with traditional files, powerful database systems, and in a mixed environment. Use of the data dictionary remains independent of the database management system, although further add-on facilities enable DATAMANAGER data definitions to be generated directly from the database data description language source coding. [Ref. 21]

The architecture, or structure, of the DATAMANAGER data dictionary is composed of four (or five) data files, called data sets in the user documentation.

The source data set contains the data definitions as originally input into the system by the user. When the user modifies or appends changes, the data definitions are automatically updated within the file.

The data entries data set contains all encoded data definitions generated by DATAMANAGER after evaluating the contents of the source data set. Data definitions are encoded to reduce the time required for DATAMANAGER to process the information within the data dictionary. During this encoding process, relationships, aliases, and classifications are also identified.

The index data set is an automated index containing the name and address of each entity definition that is in the source data or data entries data sets. The index data set

serves as a data directory to support the fastest possible retrieval of entity definitions and associated data.

The error recovery data set is used by the system as a temporary backup storage file. This capability was implemented to increase reliability by providing for automatic recovery of the dictionary contents in the case of external interruption or other system failure during a dictionary update.

The log data set is an optional capability that is highly recommended by MSP. All updating commands, associated data definitions, and amendments are logged into that file as they occur. Entries include command identification, full date, time, user, and status of all physical input/output accesses. Additionally, the data administrator has the option of specifying that all commands directed to the data dictionary be logged. When combined with other system backup facilities, this allows DATAMANAGER to be "rolled" forward from the last backup point in case full recovery is ever required.

DATAMANAGER is a powerful system that utilizes a series of interactive commands to create, maintain, and document data dictionary contents. These standard commands are listed in Table 5. DATAMANAGER provides a predefined series of standard entity-types, relationship-types, and attribute-types that form the system standard schema. These are listed in Table 6. As shown in Table 6, DATAMANAGER uses only six entity-types in the standard schema. Those elements exist within the system as members of a logical hierarchy as shown in Figure 5.1. Discussion in the user documentation reveals that DATAMANAGER strives to provide the capability to maintain all system data while maintaining ease and simplicity of logical design.

TABLE 5

DATAMANAGER Standard Commands

ADD	ALSO	ALTER
AUTHORITY	BULK	COPY
DICTIONARY	DOES	DROP
ENCODE	ENDDMR	FORMAT
GLOSSARY	INSERT	KEEP
LIST	MODIFY	PERFORM
PRINT	PROTECT	REMOVE
RENAME	REPLACE	REPORT
SHOW	STATUS	WHAT
WHICH	WHO	WHOSE

TABLE 6

DATAMANAGER Standard Schema Descriptors

PROCESS ENTITY-TYPES

MODULE	PROGRAM	SYSTEM

DATA ENTITY-TYPES

FILE	GROUP	ITEM

RELATIONSHIP-TYPES

SEE

ATTRIBUTE-TYPES

ACCESS-AUTHORITY	ADMINISTRATIVE-DATA
ALIAS	CATALOGUE
COMMENT	DESCRIPTION
EFFECTIVE-DATA	FREQUENCY
NOTE	OBSOLETE-DATE
QUERY	SECURITY-CLASS

A complete specification of the data resource of an organization requires the definition of the characteristics and of the interrelationships of data, and of the contexts in which the data is used. Accordingly, the design of DATAMANAGER provides for a hierarchy of member

types, within which it is possible to describe all elements and assemblages of data and the processess that act on the data. The number of member types defined for the basic hierarchy has been kept as small as possible while meeting these requirements. [Ref. 22]

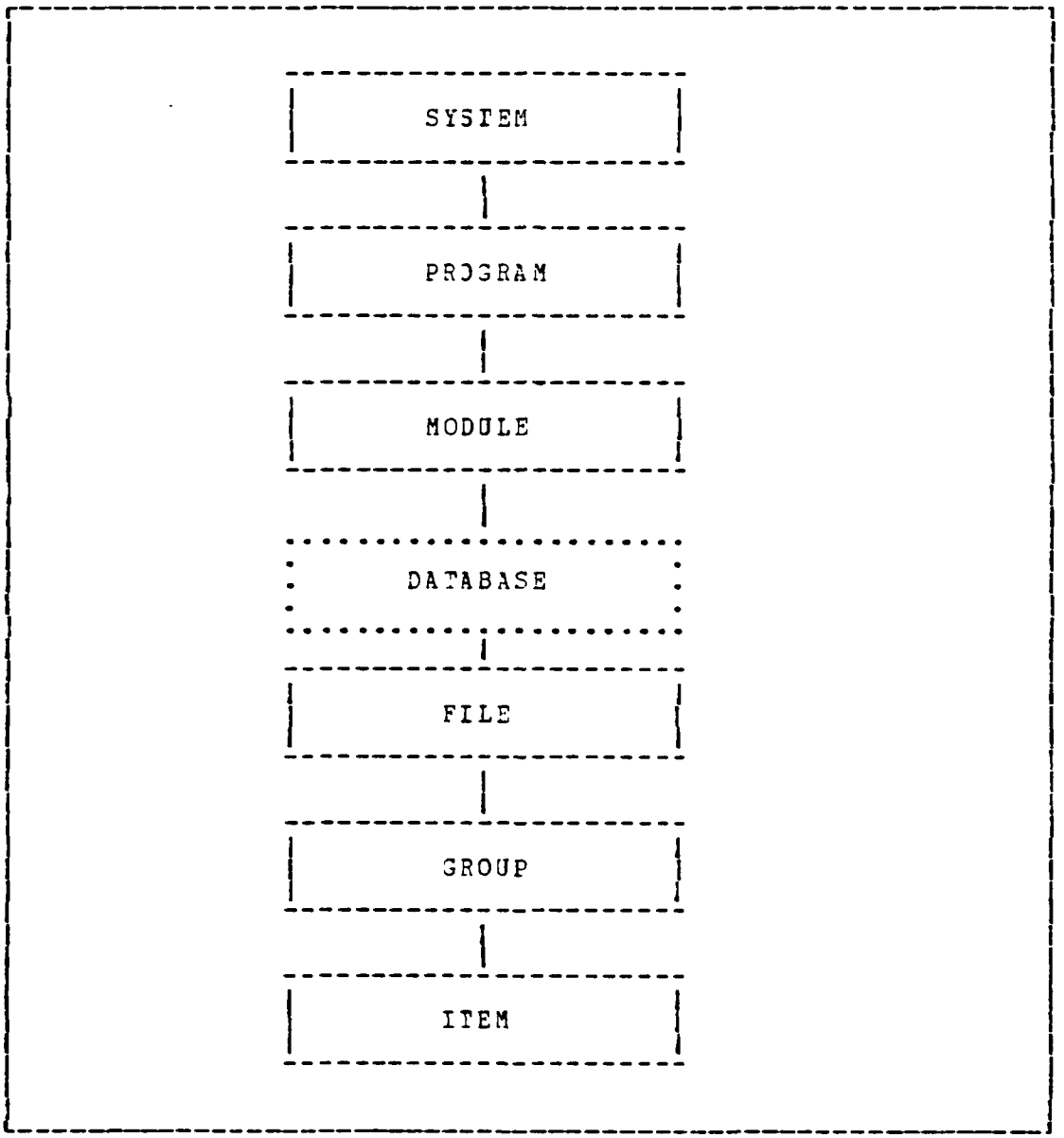


Figure 5.1 DATAMANAGER's Hierarchy of Entity-types

At the lowest level, an ITEM is a fundamental element of data, the smallest unit within DATAMANAGER. A GROUP is a collection of items or other groups. The third entity-type, the FILE, can either be implemented as a traditional file organization (a collection of data groups, independent of a DBMS), or as the equivalent association of data within a database. If DATAMANAGER is used with a database, another entity-type, DATABASE, will be provided with the database interface module, e.g., ADABAS, that is selected. The new member, in this case, ADABAS-DATABASE, will either replace the FILE element within the hierarchy, or coexist by residing between the FILE and MODULE elements. A MODULE is a collection of data that includes descriptions of a database (if used), FILEs, GROUPs, and/or ITEMs. The module is the lowest unit that can directly or indirectly manipulate data, and is a subdivision of a PROGRAM. The PROGRAM is defined in terms of collections of modules and those processes that input or output data to/from the system. A program is executable. A SYSTEM is the highest element of the DATAMANAGER hierarchy and contains all subordinate data declarations.

While DATAMANAGER stresses simplicity in the logical design of the system standard schema, it can be configured to be highly extensible. An add-on module, the User Defined Syntax Facility (UDSF), is required to support user declaration of schema descriptors. If present, this facility provides several unique capabilities. First, in addition to allowing the user to define his or her own entity-types, the module allows the data administrator to insert one (or more) of three standard sets of extended entity-types. These sets are:

1. The Extended Data Processing Structure (EDPS) which provides additional entity-types frequently used within the data processing environement. These include PROCEDURE, SUBROUTINE, and DATASET.

2. The Structured Analysis Structure (SAS) which provides entity-types frequently used when conducting structured design. These include SUBPROCESS and DATASTRUCTURE.
3. The Structured Development Structure (SDS) which strives to provide all entity-types necessary to satisfy the requirements of the majority of potential users. This collection of entity-types include all those found in the EDPS and SAS subsets.

Second, the UDSF module supports user definition of attribute-types related to both system standard and user-created entity-types. Three distinct categories of attribute-types are recognized within DATAMANAGER. These are:

1. Global (common) attribute-types which will apply to all entity-types within the structure, e.g., SECURITY-CODE.
2. Generic attribute-types which can be added to those of a specific standard entity-type, for example, FILE. Whenever a user defined entity-type is created that uses the standard entity-type's format as a base, the generic attribute-types of the standard entity-type will be passed into the new entity-type.
3. Specific attribute-types which allow the designer to tailor an entity-type to satisfy the particular requirements of that organization.

Finally, the UDSF module supports user definition of relationship-types in both forward and backward directions. This enables DATAMANAGER to support the three (or four) relationship mappings we have previously described.

Once DATAMANAGER is installed on the computer, two major steps must be conducted before information can be entered in the data dictionary. First, an empty data dictionary must

be defined using Controller commands (restricted to use by the data administrator), DICTIONARY, and AUTHORITY. Briefly, the dictionary must be created and opened, authority levels must be defined, and potential users must be identified. As the second major implementation step, member entity-types, both standard and user-created, must be be defined. Every session with DATAMANAGER is conducted as a "run", in which a series of system commands, specified by the user, are carried out. Every session must initiate with the commands DICTIONARY and AUTHORITY. After review of the user documentation, this process will probably seem difficult and confusing to most users, even to those who have worked with other data dictionaries. DATAMANAGER is, however, an impressive, powerful package in the hands of an experienced user. Our sample database, STUDENT, would be entered as a FILE (or DATABASE, if implemented). The format for an individual student's record becomes a GROUP, in which each data element, e.g., service, SSN, etc., becomes an ITEM. The structure of our example, after implementation in DATAMANAGER, would appear as shown in Figure 5.2.

DATAMANAGER aggressively supports each of the three objectives of data dictionary usage: data integrity, security, and maintenance/documentation. It enforces data integrity through its hierarchical structure of entity-types, predefined standard schema relationships, identification of aliases, and automatic update procedures. System definitions and error-checking are used to validate the structural "correctness" of each entity, relationship, and attribute as it is created or defined. Once the FILE or DATABASE is defined, DATAMANAGER monitors input of data into system structures by comparing the input to the appropriate ITEM's characteristics. Each of the MSP products, including the DBMS interfaces, displays evidence that MSP recognizes the importance of data integrity as a vital link to efficient and dependable control of data.

```
00205  PRODUCE STUDENT LAYOUTS.
00206  PRINT GIVING DESCRIPTIONS.

*****************************************************
*                DESCRIPTION OF STUDENT             *
*****************************************************
* LEVEL  NAME           LEN   TYPE   REMARKS        *
*****************************************************
*   1    STUDENT        069   GROUP  STUDENT        *
*                                                   *
*****************************************************
*   2    STUDENT-NAME   050   CHAR   50 DIG/ALPH-NUM *
*                                                   *
*****************************************************
*   2    STUDENT-SSN    011   CHAR   3 NUM,"/",     *
*                                    2 NUM,"/",4 NUM *
*****************************************************
*   2    STUDENT-SERV   005   CHAR   05 DIG/ALPH-NUM *
*                                                   *
*****************************************************
*   2    STUDENT-RANK   003   CHAR   1 CHAR,"-",     *
*                                    1 NUM          *
*****************************************************
```

Figure 5.2 STUDENT example in DATAMANAGER Structure

The DATAMANAGER nucleus provides security by inclusion of one type of security mechanism, password control. The Controller, or dictionary administrator, must assign a unique password to each authorized user. Each user and password combination must be registered within the dictionary. DATAMANAGER will reject any command session which does not commence with an AUTHORITY command followed by an authorized password.

Several additional security mechanisms can be provided by including the Audit and Security Facility module in the system implementation. First, the Controller gains the capability of registering general and specific security levels within the dictionary. Each user may be assigned a general security level in addition to the unique password previously assigned. Within the system, the Controller will assign a specific Insertion Security Level and a specific

Protection Security Level. A user whose general level is lower than the specific insertion level is not allowed to insert, modify, or delete information within the data dictionary. This provides the capability to assign "read only" access. A user whose general level is lower than the specific protection level, or one who does not have a general security level assigned, is not allowed to establish protection for system members, or data structures.

Second, users who do have a general security level equal to or higher than the specific protection level may use the PROTECT command to assign protection to specific members in the form of ACCESS, ALTER, and REMOVE security levels. This capability allows key users to control, or even prohibit, access to those structures that they own. Any member which is not owned but does require security can be assigned the same three control levels by the dictionary administrator.

Finally, the Audit module provides the capability to produce over 500 different audit reports, using information contained within DATAMANAGER. The majority of these reports are reserved for use of the dictionary administrator alone. This includes the capability of logging all commands issued to the system. This "trace" mechanism increases security by providing a record of all entries, or attempted entries, to the system.

The last significant objective of a data dictionary must be to support maintenance and documentation of the information contained within the information system. DATAMANAGER provides a set of commands unique to the maintenance function. A listing of these is shown as Table 7. Maintenance can be supported during both interactive and batch sessions. A series of query and report commands are provided with the nucleus module to support usage studies, maintenance, and documentations. These commands are listed in Table 8. The REPORT, PRINT, and GLOSSARY commands provide a great deal of

TABLE 7

DATAMANAGER Maintenance Commands

INSERT	MODIFY	ENCODE
REPLACE	BULK ENCODE	COPY
ADD	RENAME	ALTER
REMOVE	KEEP	DROP
ALSO KEEP	PERFORM	

TABLE 8

DATAMANAGER Report/Query Commands

Report Commands

PRINT	BULK REPORT	LIST
SWITCH	REPORT	SKIP
GLOSSARY	SPACE	BULK PRINT
TEXT		

Query Commands

WHAT	WHO	WHICH
WHOSE	DOES	SHOW

information to the dictionary adminsistrator and other designated users. When system data is modified, the query and report commands can be used to provide updated documentation and records.

One additional DATAMANAGER capability warrants mention with respect to maintenance and documentation. One system entity-type which has not been discussed and does not reside in the hierarchy shown earlier is the COMMAND-STREAM entity-type. This structure is a unique feature of DATAMANAGER

that allows previously stored series of commands to be executed by using the PERFORM command. The use of specific COMMAND-STREAMs can be compared to the subroutines of a general programming language. While the COMMAND-STREAM can be used in many ways within DATAMANAGER, it becomes especially useful during generation of reports and documentation during maintenance sessions. A "subroutine" can be specified that will produce all standard reports; when system information is updated, the applicable reports are produced by one simple PERFORM command at the end of the maintenance session.

C. ADR DATADICTIONARY

DATADICTIONARY is one of fourteen separate, but highly integrated, software products produced by Applied Data Research, Inc (ADR). Initially introduced in 1978, the integrated system, Relational Information Management Environment (RIME), is considered to be one of the first true examples of the fourth generation of systems software. An article in Infosystems states

> Three conditions are certain in the 1980s....First, applications packages will not meet the need for most applications that will be computerized. Second, systems software products that improved productivity, reduced application costs and increased accessibility to information in the 1970s will be even more valuable in the 1980s. And third, existing applications will not be readily rewritten or replaced and will have to be maintained for many years....The success or failure of many organizations in the 1980s will depend on how effectively they improve and integrate data processing in their operations. This is particularly critical for organizations that have been traditional data processing users over the last 20 years and have worked with second and third generation mainframe hardware and software systems. [Ref. 23]

Prior to analyzing ADR's data dictionary, it is important to review briefly the objectives of fourth generation software and integrated systems and to provide an overview of RIME.

Each of the "generations" of system software can be identified by one or more significant advancements. The first generation provided primarily assembly language programs. The second generation's gifts centered around development of high-level languages and improved operating systems. Numerous advances surfaced during the third generation, e.g., database management systems, data dictionaries, structured programming techniques, early efforts at decision support systems, and program generators. During the fourth generation, it is anticipated that advances will occur in three primary areas: very high-level languages, relational database management systems, and the automated office or integrated information center. In the latter, all automated functions, including data processing, word processing, database and file management, decision support, program development and maintenance, and communications, will be combined into one "total" system. This could, in theory, be accomplished by one giant program, or, in the case of ADR and other vendors, as a series of smaller, integrated packages.

During 1982, the U. S. Army awarded a contract for the largest, most complex information processing project ever funded by the government. Named VIABLE (Vertical Installation Automation Baseline), the project will provide a nationwide automated network that will connect forty-seven military bases to massive computer power at five regional data processing centers. The network has been designed to support the management of information in peacetime and in times of war and other national emergencies. During the planning period, interest centered on three principal functional areas: communication, interactive program development, and database management. The primary contractor, Electronic Data Systems, selected 11 of ADR's products for use as the base of the VIABLE system. A complete list of ADR/RIME elements is included as Table 9.

TABLE 9

Components of ADR's DATACOM System

	Component	Function
	DATACOM/DB	Relational Database System
	DATA DICTIONARY	Resource Control System
	DATAQUERY	English-like Query Language
	DATAREPORTER	Info. Retrieval/Reporting
	DATAENTRY	On-line Data Entry System
	COBOL/DL	Extended Language/Utilities
	LIBRARIAN	Program Management System
	ROSCOE	Program Maintenance System
	LOOK	Real-time Measurement System
	METACOBOL	Language Pre-compiler
	AUTOFLOW II	System Development Tool
*	ADR/D-NET	Distributed Database Network
*	ADR/EMAIL	Electronic Mail System
*	ADR/IDEAL	Interactive Develop. System

* New ADR products which have not yet been included in the Army's VIABLE project.

Some of these elements can be considered to be high-priced extras or application-specialized options. If an organization were to utilize all components, users would have access to a complete database system with data dictionary, a relational query language, report and graph generators, extended COBOL compiler, program development support, distributed local data network, electronic mail system, and more.

According to ADR literature, the heart of the integrated system is DATADICTIONARY. The company's database system, DATACOM/DB, a true relational database[3] system that utilizes a patented flexible data structure, was designed especially to interact with DATADICTIONARY. As an active dictionary

---

[3]A relational database is one in which the relationships between data are implied by the values of the data. For example, two records are related if they have the same attribute, as STUDENT and PROFESSOR are related by the fact that they are associated with a particular CLASS.

system, DATADICTIONARY is queried by all other components of the system prior to access of system information. This maximizes data integrity while minimizing data redundancy. DATADICTIONARY offers a menu-driven user interface. It provides security, supplies full documentation/maintenance capabilities, and can be extended to interact with future system products and to support future user requirements. The documentation provided with DATADICTIONARY and other ADR packages is almost overwhelming in its completeness. The dictionary alone has fifteen separate volumes. While an extremely capable system, DATADICTIONARY is not one that will be easily or quickly mastered.

DATADICTIONARY provides 20 standard entity-types in its system standard schema and supports user creation of additional, more application-specific schema descriptors. For most applications, the standard types listed in Table 10

TABLE 10

ADR DATADICTIONARY Standard Entity-types

DATABASE	KEY	SYSTEM	REPORT
AREA	ELEMENT	PROGRAM	JOB
FILE	LIBRARY	MODULE	STEP
RECORD	MEMBER	DATAVIEW	AUTHORIZATION
FIELD	PANEL	PERSON	NODE

will prove to be sufficient. DATADICTIONARY maintains a logical hierarchy among the principle standard entity-types, as indicated in Figure 5.3. Many of the standard entity-types are provided with primary relationships already defined with key subordinate entity-types. For example, in our STUDENT example, we will initially use the entity-type

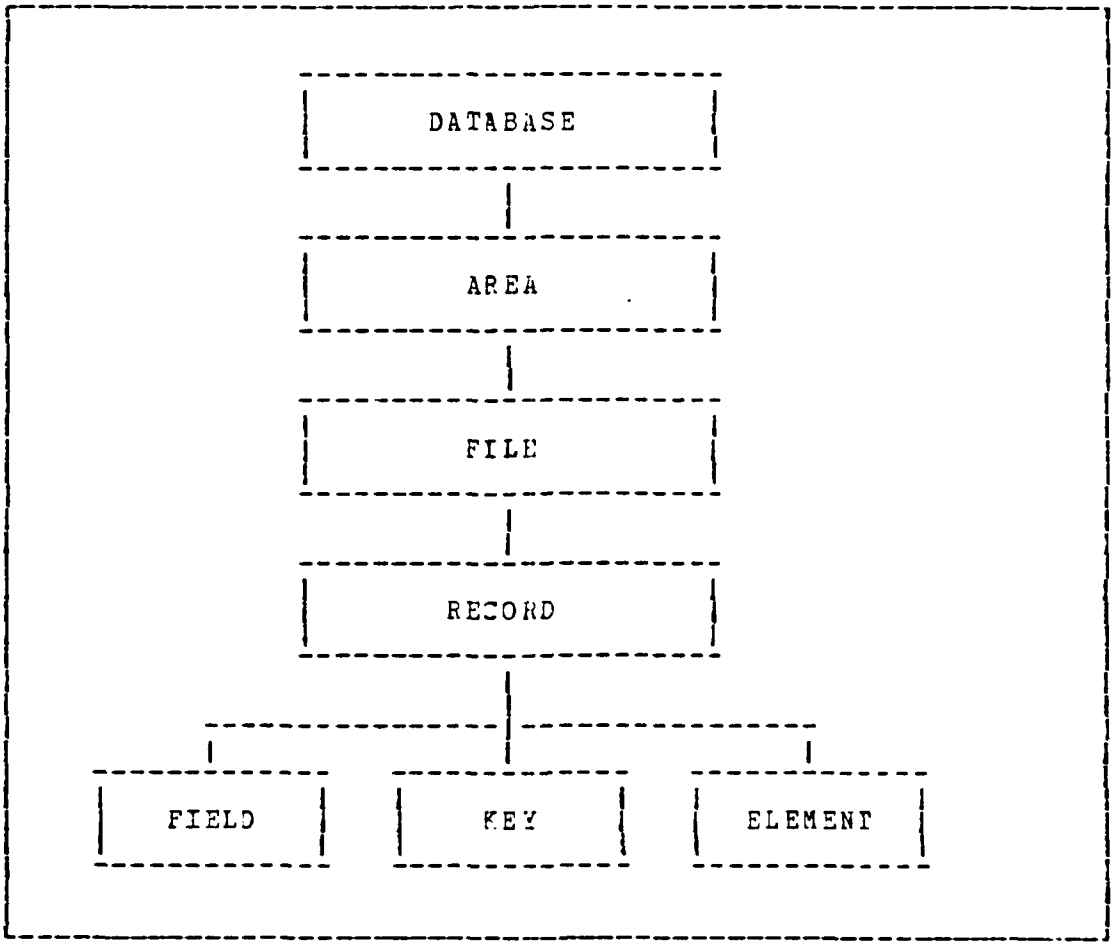


Figure 5.3 A Logical Hierarchy of Entity-types

DATABASE to create our sample database. When we define the database entity-occurrence, the DATABASE-AREA relationship is automatically provided. Similarly, when the area-occurrence is defined, the AREA-FILE relationship is established by DATADICTIONARY. In the case of RECORD, creation of an occurrence provides three relationship-types: RECORD-FIELD, RECORD-KEY, and RECORD-ELEMENT. These three relationships, at the lowest level of the logical hierarchy, support actual entry of attribute-values, or data. Whether system-defined or user-created, all relationship-types in DATADICTIONARY have four attributes:

1. Relationship-mapping -- describes the number of entity-occurrences which are the subjects and the objects of this relationship, e.g. the type of the relationship. DATADICTIONARY supports four types of relationship mappings, i.e. one-to-one, one-to-many, many-to-one, and many-to-many.
2. Required-relationship -- describes whether each entity-occurrence in the named object entity-type is to be related to at least one entity-occurrence of the named subject entity-type.
3. Automatic-relationship - describes waether each entity-occurrence of the named object entity-type is to be automatically related to an entity-occurrence of the named subject entity-type when the object is added.
4. Ordered-relationship - describes whether the order of relationships added in this relationship-type is significant. An ordered-relationship allows entity-occurrences to be retrieved and displayed in a specific order.

If using the interactive version, DATADICTIONARY Online, the user will be prompted by a series of panels, or menus. The Master Menu is displayed in Figure 5.4. The master menu supports creation, modification, and deletion of entity-occurrences. Additionally, it provides access to all other system menus through option (7). The following procedures would be utilized to create the STUDENT example within DATADICTIONARY. First, the Add Detail routine, option (2), is selected. In answering the system prompts, the user creates the new entity-occurrence, DATABASE.STUDENT in the following dialog:

```
=>_____________________________________________________

DDOL: SELECTION CRITERIA FOR DETAIL ADD
LV  TY  ENTITY    RECORD   DD OCCURRENCE NAME   VER  STAT
00   E  DATABASE           STUDENT              001

          CURRENT OCCURRENCE     QUALIFIER:
       DATABASE              STUDENT              (001) TEST
**********************************************************
                    DETAIL ADD
 ATTRIBUTE       VALUE

 DESCRIPTION     NPS STUDENT DATABASE
 CONTROLLER      DEPARTMENT OF REGISTRAR
 AUTHOR          REGISTRAR
 BASE-ID         001
 BASE-TYPE       ADR/DB
 DBMS-USED       RELATIONAL
```

```
=>_____________________________________________

MASTER MENU

ENTER THE REQUESTED OPTION ==> THERE ARE 08 OPTIONS

  1. DISPLAY MENU    MENU FOR DISPLAY FUNCTIONS
  2. ADD DETAIL      ADD DETAIL ENTITY-OCCURRENCE
  3. DELETE DETAIL   DELETE DETAIL ENTITY-OCCURRENCE
  4. UPDATE DETAIL   UPDATE DETAIL ENTITY-OCCURRENCE
  5. COPY            COPY/MODEL ENTITY-OCCURRENCE
  6. STATUS CHANGE   CHANGE ENTITY-OCCURRENCE STATUS
  7. SUPPORT MENU    ALIAS, DESCRIPTOR, RELATIONSHIP,
                     TEXT, AND QLIST
  8. SECURITY        OCCURRENCE SECURITY MAINTENANCE
```

Figure 5.4 ADR DATADICTIONARY Master Menu

Each of the 20 standard entity-types will contain predefined key attributes. Values for these attribute-types are entered during the Add Detail routine. In the case of the DATABASE entity-type, and as was shown above, the key attributes are DESCRIPTION, CONTROLLER, AUTHOR, BASE-ID, BASE-TYPE, and DBMS-USED.

In similar fashion, the user must create the subordinate logical structures, AREA.STUDENT, FILE.STUDENT, and RECORD.STUDENT. As each occurrence is created, it must be

related to the next highest entity-occurrence in the logical hierarchy, e.g., FILE.STUDENT must be related to AREA.STUDENT. For this process, the user invokes the Relationship Definition Panel to define the relationships. DATADICTIONARY will respond with the Relationship Definition Display which presents the characteristics of each of the relationships as it is enacted. Examples of these panels are shown below:

```
=>_____________________________________________________

DDOL:  RELATIONSHIP DEFINITION
RELATIONSHIP NAME          $INTERNAL
SUBJECT ENTITY TYPE        DATABASE.STUDENT
OBJECT ENTITY TYPE         AREA.STUDENT

=>_____________________________________________________

              RELATIONSHIP DEFINITION DISPLAY

SELECTION:
$INTERNAL    DATABASE.STUDENT     AREA.STUDENT
NAME         SUBJ TYPE  OBJ TYPE  MAP  REQ  AUTO  ORDER
$INTERNAL    DATEBASE     AREA     1M   Y    N     N
```

As a final step in installing the STUDENT database, QLIST commands must be used to define specific fields, keys, and elements within RECORD.STUDENT. This is the point where the specific attributes of the STUDENT example, e.g., SSN, Name, Service, and Rank, are entered into the database design. The user defines attribute name, parent, class, type, length, and number of repetitions. One example of this process is as follows:

```
=>_____________________________________________________

DDOL: SELECTION CRITERIA FOR RECORD QLIST MAINT
LV  TY  ENTITY   RECORD  DD OCCURRENCE NAME  VER  STAT
00   E  RECORD           STUDENT                 TEST

          CURRENT OCCURRENCE     QUALIFIER:
        RECORD               STUDENT              (001) TEST
**********************************************************
                RECORD QLIST MAINTENANCE

 E FC FIELD NAME  PARENT NAME  INSERT AFT  C T LEN REP
   A  SERVICE     SSN          NAME        S C 004 001
```

Looking at the last line of the figure, the user has indicated the following:

FC (function code) = Add a field
Field Name = SERVICE
Parent Name = SSN (in this case, this is the Key field)
Insert After = NAME (NUMBER's value will follow NAME's)
C (Class) = Simple (as opposed to a compound field)
T (Type) = Character (vice a numeric or binary field)
LEN (Length of Field) = 4
REP (Number of Repetitions) = 001 (vice a repeating field)

At this point, the schema of STUDENT has been entered into DATADICTIONARY. The user may now use DATACOM/DB facilities to enter attribute-values into the system. Upon completion, the database administrator or authorized users can create as many external views, or subschemas, as desired.

DATADICTIONARY receives high marks in the areas of data integrity, security, and documentation/maintenance. DATADICTIONARY's logical hierarchy of structures and systematic installation procedures tend to enforce data integrity. The dictionary's extension routines and view generation processes have been written to ensure that data integrity is maintained throughout expansion or specialization of the database. To enforce security, DATADICTIONARY provides multiple layers of protection. Two separate and independent mechanisms are provided in all implementations. These are (1) use of entity passwords, and (2) inclusion of locks and override codes. If the installation is the Online version, a third mechanism, user validation, is available. As each entity is created, or at any time afterwards, a four-digit password can be assigned to that entity. Passwords can be either unique or assigned to a series of related entities. Any user attempting to modify or access a password-protected entity-occurrence will be queried to provide the applicable password prior to gaining access. The second layer of protection centers on use of LOCK and OVERRIDE codes. Unlike passwords, which either allow or prohibit access, lock codes can be utilized to limit the <u>degree</u> of access granted. Three levels of security are provided:

LOCK0	No restrictions exist on an entity (default setting)
LOCK1	The entity cannot be updated or deleted without an override code. The entity can be copied, displayed, or printed without restrictions.
LOCK2	No action will be permitted unless the override code is given to the system.

The actual override codes will be used dictionary-wide, that is, a single code will exist to satisfy LOCK1 conditions while another code exists to access entities protected by LOCK2. Finally, if using DATADICTIONARY Online, the highest layer of security becomes user validation. The name of each user of the system is defined as a PERSON entity-type. Each entity-occurrence will include a unique password which must be provided to enter the system through the online interface. Four levels of authorization are supported by DATADICTIONARY:

_DIS	The user is allowed to display all data in the dictionary.
_UPD	The user is allowed to update the dictionary.
_COP	The user is allowed to copy an entity.
_ADM	The user is allowed the use of all commands and is allowed to process all panels.

Authorization at one level will automatically provide all lower authorizations.

ADR's multiple-layered approach to security provides a system that is both highly flexible and very secure. The database administrator will be able to provide whatever degree of access that is required to each individual user as well as to each group of users within the system. If one layer of security is broken, access will be prevented by the other security mechanisms.

> Invocation of any function thus authorized on any entity is still subject to the password and lock provision discussed earlier in this section. Thus, a user with DDD_UPD authorization cannot modify an entity that is password protected unless the required password is supplied. [Ref. 24]

DATADICTIONARY provides extensive capabilites to support maintenance and documentation of the data dictionary. It can be maintained by using either the Online maintenance facility or available batch commands. If using the online facility, a series of screen panels will again guide the user through the desired maintenance activity. This facility will greatly enhance individual changes, however, major changes affecting numerous entities would be initiated most easily through batch commands. In either case, maintenance centers around four principal functions:

1. adding, copying, updating, or deleting system entities
2. search for, identification of, and creation of entity aliases
3. maintenance of descriptors and schema descriptors
4. maintenance of descriptive tests associated with system entities

Similarly, DATADICTIONARY provides numerous report generation capabilities, most of which can be initiated through either batch or Online Maintenance sessions. Principal report types are shown in Table 11. Generated reports will support both the initial generation of user databases and subsequent maintenance of system data and the structures utilized to display it.

TABLE 11

Principal Reports of DATADICTIONARY

INDEX	INDENTED	DETAIL
FIELD	TEXT	ALIAS
DESCRIPTOR	RELATIONSHIPS	DEFINITIONS

D. ORACLE

ORACLE is a relational database management system developed by Relational Software Incorporated of Menlo Park, California. It was originally developed for use with Digital Equipment Corporation PDP minicomputers and has been converted to operate on IBM mainframes as well [Ref. 25]. Included in ORACLE is a dependent data dictionary that performs a limited number of the functions discussed in previous chapters.

Data is stored in ORACLE as relations, or two-dimensional tables, which are organized into rows and columns. SQL (System Query Language) is used for query, manipulation, definition, and control of the ORACLE database. Information about the contents of a table, its creator, authorized users, calling programs, and associated views is kept in the data dictionary and can be retrieved via SQL commands.

ORACLE's logical hierarchy of structures, as shown in Figure 5.5, demonstrates the comparative simplicity of this system. In this figure, a single arrowhead represents a one-to-one relationship while the double arrowheads signify one-to-many relationships. The database is divided into logical partitions which can only be created or altered by the database administrator. When users define tables, the system allocates memory for one indexspace and one dataspace. The indexspace is used by the database/dictionary to store information about the table while the dataspace is utilized for storing the actual information. As data is entered into the database, the system automatically appends extents (and pages) as necessary to support specific tables.

ORACLE's 18 data dictionary tables are described in Figure 5.6. An example of one of the tables, CATALOG, appears in Figure 5.7. Tables with the "SYS" prefix include

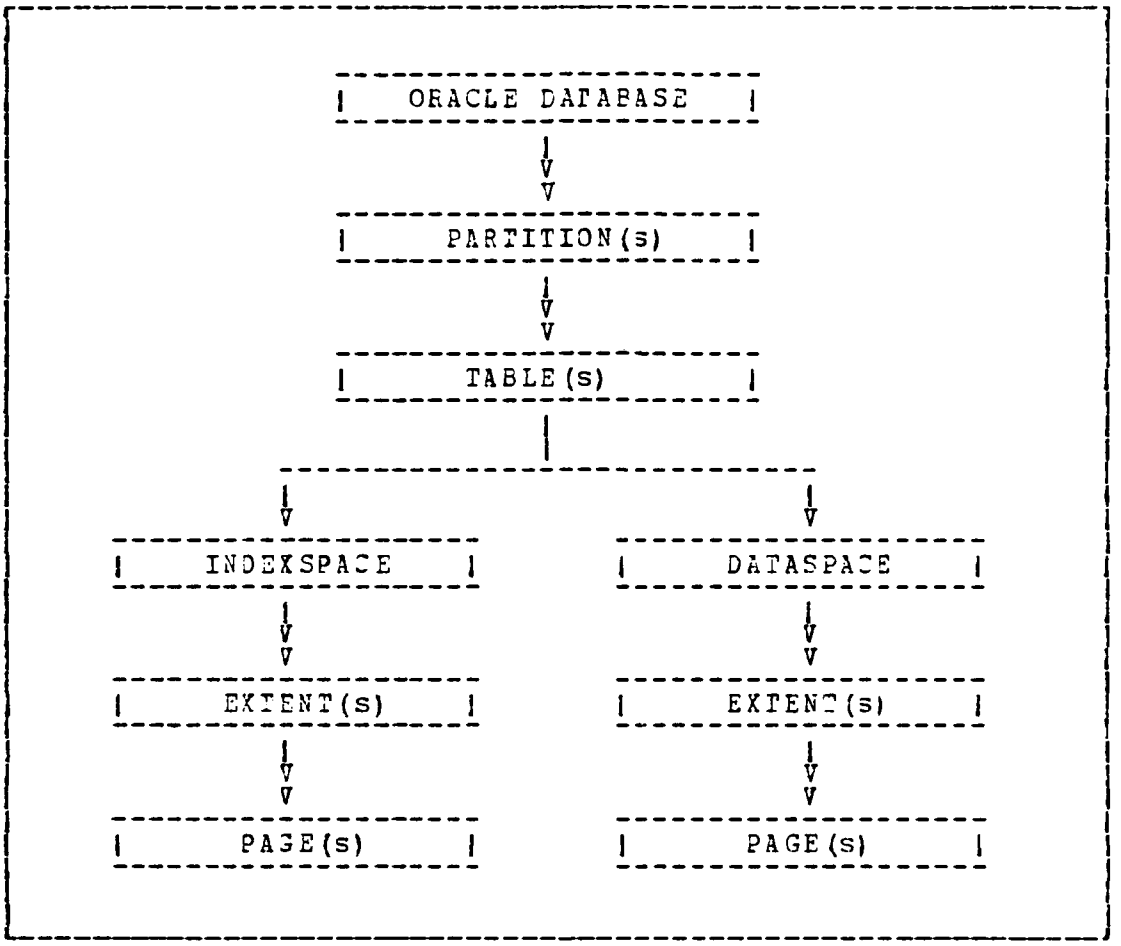


Figure 5.5 ORACLE's Logical Hierarchy

information on system data in addition to the user's data. For example, a display of SYSCATALOG might appear as Figure 5.3. In this particular example, there are 20 entries, 18 of which are system tables or views.

ORACLE's data dictionary is automatically updated whenever any additions or deletions are made to the database or when views are defined or user privileges are changed, so it always has a current description of the database. As an example, assume a new view, NAVYVIEW, is created using the SQL CREATE command:

```
JFI> CREATE VIEW NAVYVIEW AS
  2          SELECT NAME,SSN,RANK
```

```
DTAB
  - Description of tables & views in Oracle Data
    dictionary
SYSCATALOG
  - Profile of tables & views accessible to user
CATALOG
  - Profile of tables accessible to user, excluding
    data dictionary
TAB
  - List of tables, views, clusters, and synonymns
    created by user
SYSCOLUMNS
  - Specifications of columns in accessible tables
    and views
COLUMNS
  - Specifications of columns in tables (excluding
    data dictionary)
COL
  - Specifications of columns in tables created
    by the user
SYSINDEXES
  - List of indexes, underlying columns, creator,
    and options
INDEXES
  - Indexes created by user & indexes on tables
    created by user
SPACES
  - Selection of space definitions for creating
    tables & clusters
VIEWS
  - Quotations of the SQL statements upon which
    views are based
SYSTABAUTH
  - Directory of access authorization granted by
    or to the user
EXTENTS
  - Data structure of extents within tables
STORAGE
  - Data and Index storage allocations for user's
    own tables
SYSSTORAGE
  - Summary of all database storage -- for DBA
    use only
SYSUSERAUTH
  - Master list of Oracle users -- for DBA use only
SYSEXTENTS
  - Data structure of tables throughout system
    -- for DBA use only
PARTITIONS
  - File structure of files within partitions
    -- for DBA use only
```

Figure 5.6 Tables of the ORACLE Data Dictionary

```
3          FROM STUDENTS
4          WHERE SERVICE = "USN"

View created.
```

NAME	CREATOR	TABTYPE	TABID
STUDENTS	LANDIN	TABLE	228609
ARMYVIEW	OWENS	VIEW	268800

Figure 5.7 ORACLE CATALOG Listing

NAME	CREATOR	TABTYPE	TABID
HELP	SYSTEM	TABLE	9985
DUAL	SYSTEM	TABLE	10497
STORAGE	SYSTEM	VIEW	11520
EXTENTS	SYSTEM	VIEW	11776
SPACES	SYSTEM	VIEW	12288
SYSCOLUMNS	SYSTEM	VIEW	12544
COLUMNS	SYSTEM	VIEW	12800
SYSCATALOG	SYSTEM	VIEW	13056
CATALOG	SYSTEM	VIEW	13312
SYSINDEXES	SYSTEM	VIEW	13568
INDEXES	SYSTEM	VIEW	13824
VIEWS	SYSTEM	VIEW	14080
SYSAUTH	SYSTEM	VIEW	14336
TAB	SYSTEM	VIEW	14848
COL	SYSTEM	VIEW	15104
EXPTAB	SYSTEM	VIEW	15360
EXPVEW	SYSTEM	VIEW	15616
DTAB	SYSTEM	TABLE	15873
STUDENTS	LANDIN	TABLE	228609
ARMYVIEW	OWENS	VIEW	268800

Figure 5.8 ORACLE SYSCATALOG Listing

Upon completion of this dialog, all ORACLE data dictionary files will have been automatically updated to include the new view. The CATALOG table would now appear as shown in Figure 5.9.

ORACLE provides security by using its data dictionary to control access within the database. The database administrator (DBA) provides the first level of access by entering the user's name into the data dictionary's SYSUSERAUTH

NAME	CREATOR	TABTYPE	TABID
STUDENTS	LANDIN	TABLE	228609
ARMYVIEW	OWENS	VIEW	268800
NAVYVIEW	LANDIN	VIEW	288240

Figure 5.9 ORACLE CATALOG Listing With New View

table. Initial privileges, or subsequent changes to authorized privileges, are issued using the GRANT or REVOKE commands. ORACLE also supports multi-layered access: in addition to privileges authorized by the DBA, a user can grant various degrees of access privilege to others for tables or views which he or she has created. A list of current authorizations is maintained in the dictionary's SYSTABAUTH view, as shown in Figure 5.10.

ORACLE is a strong performer in the data integrity category. Since the data dictionary is an integral part of the database system, data is only maintained at one location within the database. This prevents two users from acquiring data from the database and getting different results. If data were duplicated within the system, it would be possible for one location to be updated while the other was not. Figures 5.7 through 5.10 show that the ORACLE user will deal mostly with subsets of the database, or subschemas.

ORACLE's documentation is limited to the information that can be found in the data dictionary tables. It does not provide information about which users use which data, how often data is used, or when it is used. ORACLE does support maintainability through automatic update of its tables and through the concept of data independence. This concept implies a separation of data definitions from the programs or queries that might access the data in the

GRANTOR	GRANTEE	TABTYPE	AUTHORITY
CREATOR	TNAME		
SYSTEM	PUBLIC		
SYSTEM	HELP	TABLE	SELECT
SYSTEM	PUBLIC		
SYSTEM	DUAL	TABLE	SELECT
SYSTEM	PUBLIC		
SYSTEM	SYSCOLUMNS	VIEW	SELECT
SYSTEM	PUBLIC		
SYSTEM	COLUMNS	VIEW	SELECT
SYSTEM	PUBLIC		
SYSTEM	SYSCATALOG	VIEW	SELECT
SYSTEM	PUBLIC		
SYSTEM	CATALOG	VIEW	SELECT
SYSTEM	PUBLIC		
SYSTEM	SYSINDEXES	VIEW	SELECT
SYSTEM	PUBLIC		
SYSTEM	SYSTABAUTH	VIEW	SELECT
SYSTEM	PUBLIC		
SYSTEM	TAB	VIEW	SELECT
SYSTEM	PUBLIC		
SYSTEM	EXPTAB	VIEW	SELECT
SYSTEM	PUBLIC		
SYSTEM	EXPVIEW	VIEW	SELECT
SYSTEM	PUBLIC		
SYSTEM	DTAB	VIEW	SELECT
LANDIN	OWENS		
LANDIN	STUDENTS	TABLE	SELECT
LANDIN	OWENS		
LANDIN	STUDENTS	TABLE	DELETE
LANDIN	OWENS		
LANDIN	STUDENTS	TABLE	UPDATE
OWENS	OWENS		
OWENS	ARMYVIEW	VIEW	DROP
OWENS	OWENS		
OWENS	ARMYVIEW	VIEW	SELECT
LANDIN	OWENS		
LANDIN	NAVYVIEW	VIEW	SELECT

Figure 5.10 ORACLE SYSTABAUTH Listing for User Owens

database. This allows the structures or definitions of the data constructs to be modified without necessitating changes in the programs or queries that access the database. If a table is extensively modified, a view can be created to interface with current programs. ORACLE's data integrity will maintain the currency of the view by automatically updating the view whenever applicable portions of the governing table are modified.

ORACLE does provide the basic functions of definition, update, retrieval, and software interface. However, like other relational database management systems with dependent data dictionaries, it does not offer the range of functions of the other data dictionaries discussed in this chapter, nor does it accomplish satisfactorily the three main objectives of data management discussed in Chapter IV. ORACLE's data dictionary

> provides little more than a method of defining the schema. The relational database management system 'dictionary' arises because the system needs a way to store the schema and it does this through the use of the same tables (relations) as it uses for the main database. [Ref. 26].

ORACLE could, however, serve as a good starting point for further development.

> The modern relational DBMS does provide a very good basis for a good dictionary system. This is because the normal relational DBMS is equipped with two features that help in making the implementation easy:
> 1. Many relational DBMS now have a "triggering" feature that causes a procedure to be invoked on some data condition or event. Such a feature is needed to tie a DBMS to a dictionary system.
> 2. The availability of the schema tables substantially reduces the effort in implementing the dictionary system. [Ref. 27]

The most important shortcoming of ORACLE's data dictionary is its lack of documentation, without which it is difficult

to manage all aspects of an organization's data. If this objective were incorporated into the system, ORACLE would be a much more valuable tool.

E. COMPARISON OF DATA DESIGNER, DATAMANAGER, DATADICTIONARY, AND ORACLE

Now that four representative samples of commercial data dictionaries have been evaluated, we will compare the primary features of each and identify which one(s) have come closest to providing the features of our ideal system. For ease of comparison, we have grouped all of the features, functions, and guidelines that have been identified into the six evaluation criteria categories: system standard schema & extensibility, command and query languages, ease of use (including menus), security, documentation and reports, and application interfaces.

As the data dictionaries are evaluated in each of the six categories, a brief chart will be used to compare each dictionary against the FIPS standards. Each chart will compare five data dictionaries:

FIPS = The ideal/FIPS data dictionary
MSP = MSP DATAMANAGER
ADR = ADR DATADICTIONARY
DDE = DATA DESIGNER
ORA = ORACLE DBMS/DD

A very subjective scoring system will be used, with grades ranging from three to zero. The ideal/FIPS standard will automatically receive a grade of "3" in each area, representing the ideal combination of features. The meaning of each grade is as follows:

"3" = Very strong performance by DD; no criticism
"2" = Good performance by DD; one or more significant shortcomings
"1" = (1) DD supports functional area very poorly;
(2) DD does not support functional area, but another component of the system does.
"0" = DD (and remainder of system) fails to support this function

First, the data dictionary should provide a system standard schema and the capability to add new entities, relationships, and attributes to it. As shown in Table 12, while DATADICTIONARY and DATAMANAGER closely resemble the ideal system proposed by the FIPS, DATA DESIGNER and, in particular, ORACLE fail to provide these capabilities. DATAMANAGER supports three "add-on" collections of schema descriptors. When added to the standard schema, each will increase DATAMANAGER's capabilites to support a specific application, e.g., programming.

TABLE 12

Category One: Schemas and Extensibility

Functional Category	FIPS	MSP	ADR	DDE	ORA
System Stand. Schema	3	3	3	1	0
Entity-types	(10)	(7)	(20)	(2)	(?)
Relationship-types	(7)	(1)	(10)	(1)	(?)
Attribute-types	(55)	(13)	(50+)	(?)	(?)
DA/User Extensible	3	3	3	0	0
Category Subtotals	6	6	6	1	0

Second, the data dictionary should provide a command language that will support queries from users while reserving some capabilities solely for the use of the dictionary administrator. This last ingredient supports security and data integrity. Again, as seen in Table 13, DATADICTIONARY and DATAMANAGER provide all capabilities of the FIPS standard while the other two lag behind.

TABLE 13

Category Two: Command/Query Languages

Functional Category	FIPS	MSP	ADR	DDE	ORA
CMD Interface Lang.	3	3	3	3	2
Query Commands	3	3	3	1	1
DA-Only Commands	3	3	3	0	2
Category Subtotals	9	9	9	4	5

Third, the ideal data dictionary must be relatively easy to use, yet still powerful enough to support the experienced user. One of the major ingredients of user-friendliness is a menu-driven (or panel-driven) format. Good, easy-to-understand examples are another important aid to the new user. Table 14 reveals that, in our opinion, none of the four systems can be considered easy to use. Looking at the four as a group, two fail to use menus, one provides examples which are complex and hard to understand, and the fourth fails to provide either menus or good examples.

Fourth, security is one of the primary objectives of a data dictionary. It should not only be able to control general access to the system, but should also support the capability to provide different levels of access to different users. In Table 15, three of the four, DATADICTIONARY, DATAMANAGER, and ORACLE receive high marks for providing both aspects of security. Security for information contained within DATA DESIGNER must be provided by the parent DBMS.

Fifth, the clearness and logical layout of system documentation should be considered. Additionally, the reports

TABLE 14

Category Three: Relative Ease of Use

Functional Category	FIPS	MSP	ADR	DDE	ORA
Menu-Driven	3	0	3	0	0
New User Friendly	3	1	2	3	2
Good Setup Example in Documentation	3	1	2	3	3
Category Subtotals	9	2	7	6	5

TABLE 15

Category Four: Security

Functional Category	FIPS	MSP	ADR	DDE	ORA
Access Control (Password)	3	3	3	1	2
Degrees of Access (Levels)	3	3	3	1	3
DA-only Privileges	3	3	3	2	3
Category Subtotals	9	9	9	4	8

and the documentation prepared by the data dictionary must be evaluated for usability. As indicated in Table 16, each of the four data dictionaries approaches that of our ideal FIPS standard. It is interesting to note that the two frontrunners, DATADICTIONARY and DATAMANAGER, have some problems with documentation complexity.

TABLE 16

Category Five: Documentation and Reports

Functional Category	FIPS	MSP	ADR	DDE	ORA
SYS Documentation clear/laid out well	3	2	2	3	3
Good Examples of Report Types	3	2	3	3	3
Reports Readable	3	3	3	3	3
Category Subtotals	9	7	8	9	9

Finally, the ideal data dictionary should support a variety of applications, interfacing with both DBMS and programming languages. DATADESIGNER and DATAMANAGER both provide interfaces to one or more DBMS and to two or more programming languages. Table 17 pertains. While DATA DESIGNER and ORACLE only interact with their system DBMS, DATAMANAGER provides flexibility and versatility by supporting several popular DBMS.

TABLE 17

Category Six: Application Interfaces

Functional Category	FIPS	MSP	ADR	DDE	ORA
DBMS Interface(s)	3	3	2	1	1
Language Interfaces	3	3	3	1	1
Category Subtotals	6	6	5	2	2

When total "scores" are calculated, the results are as shown in Table 18. While none of the systems provides all of the characteristics of the ideal/FIPS system, ADR DATADICTIONARY and MSP DATAMANAGER come the closest. If an organization were starting "fresh", with no previous invest-

TABLE 18

Data Dictionary Comparison Totals

Functional Category	FIPS	MSP	ADR	DDE	ORA
Schemas/Extensible	6	6	6	1	0
Command/Query Lang.	9	9	9	4	5
Ease-of-Use	9	2	7	5	5
Security	9	9	9	4	8
Documentation/Rpts	9	7	8	9	9
Application Inter.	6	6	5	2	2
Comparison Totals	48	39	44	25	29

ment in software, the ADR family of products, RIME, warrants serious consideration. If, on the other hand, the organization already has one of the popular DBMS, and is simply seeking to add a new, or better, data dictionary, the free-standing DATAMANAGER might very well satisfy the need. In each of these two excellent commercial packages, the observed shortcomings lie in the areas of user friendliness and clear examples for new users. Although important requirements, these faults will be overcome as the users gain experience.

In the case of the other two dictionaries, their shortcomings would be far harder to forgive. Their problems lie

in areas of standard schemas, extensibility, security, etc. Each seems more user-friendly, but, since they do less, there are fewer procedures to be explained. DATA DESIGNER, although an interesting package, simply does not provide several of the primary characteristics that we expect to find in an ideal data dictionary. ORACLE is certainly the weakest of the four dictionaries we evaluated. As part of the ORACLE DBMS, this system does provide some data dictionary features. However, it is not the full-featured data dictionary we would recommend.

VI. EXPANSIONS OF THE ROLE OF DATA DICTIONARIES

In this chapter we will suggest ways in which the role of the data dictionary can be expanded beyond the basic uses discussed in previous chapters. We will look first at how the data dictionary can enforce standards in today's increasingly common distributed data processing environment. Then we will show how the process of decision making can be supported through the use of a data dictionary. In conclusion, we will attempt to foresee where data dictionary technology will lead information resource management in the years to come.

A. DISTRIBUTED DATA PROCESSING

Our discussion of databases up to this point has centered around the assumption that an organization has one centralized database, with centralized database management and control, that would be accessed by all users. However, many organizations have decided to distribute computing power to various departments and/or outlying sites, depending on the organization's structure. In such a situation, it is also likely that the organization's database will have to be distributed. A distributed database is "a consistent, logically interrelated collection of data stored at dispersed locations" [Ref. 28]. These dispersed locations, called nodes, are connected by means of a network which allows the nodes to communicate.

Many factors have contributed to the increasing popularity of distributed processing. Two of the most important are the following: [Ref. 29]

1. Numerous advances in technology that have provided more powerful processing hardware at lower cost and improved communication and network capabilities.
2. The need for faster and easier access to time-critical information to assist in the decision making of organizations with geographically dispersed components requiring unified information sharing and processing. (This concept will be discussed in detail in the next section.)

For organizations that employ a centralized approach to control widely-dispersed, autonomous divisions, an attempt to adhere to the traditional concepts of centralized information resources may be ineffective and self-defeating. These organizations might be tempted to sacrifice the ability to better satisfy user needs in order to preserve control and traditional relationships. Fortunately, managers are rapidly becoming aware of the many potential advantages of distributing some, or all, of the organization's data processing functions to the user level. Technological advances continue to encourage these changes because

> The availability of major computing resources in small, low-cost packages allows the dedication and distribution of needed capabilities, either standing alone or interconnected, when and where they are needed. Many of the complexities of centralized large-scale computing facilities are no longer necessary. [Ref. 30]

It is important to remember, however, that

> the complexities of integrated systems require digital data communications, appropriate software, and extensive planning and coordination. These complexities should not be underestimated. [Ref. 30]

One very successful corporation, Hewlett-Packard, utilizes a combination of centralized, decentralized, and

distributed systems to support a variety of needs within the organization. Corporate planning, employee benefits, and establishment of standards are performed on mainframes located at central management. Daily operations, data processing, and employee pay and records have been decentralized and are independently performed by each division. Other functions, e.g., customer sales and support, have been distributed to increase responsiveness and timeliness.

> Successful systems put the control of the data close to the source of the information and the control of processing close to the manager responsible for the function being performed. In an organization like Hewlett-Packard, this will frequently, but not always, imply distributing the processing. Distributed processing has made it possible for us to adapt to a constantly expanding geographic operation, and a constantly changing organizational structure, while maintaining consistent administrative support. [Ref. 31]

Another class of organization includes those that have become so large and dispersed that they simply cannot be supported effectively by totally centralized resources. The armed services are prime examples of this type. For example,

> In an organization as large and decentralized as the Navy, it would be impossible and inappropriate to impose centralized control over the thousands of individual small system applications that are clearly being put to productive use. In fact, their main strength is their ability to solve many of the information-handling problems of users at the local level, without the need for centralized software development and procurement delays. [Ref. 32]

In the years ahead, a growing awareness of these conditions will drive an ever-increasing number of military organizations to distribute some portion of their information resource needs.

Data dictionaries that are designed for operations within distributed environments will require all of the

capabilites of those operating solely in a centralized environment. However, a distributed data dictionary must support three specialized functions in addition to basic data dictionary functions:

1. the ability to locate data within the network
2. the coordination/management of distributed data
3. the ability to perform data transformation in support of user applications

The distributed data dictionary's directory function enables it to identify which network node contains the specific information that is needed. Whether the particular database is distributed by replication or partitioning, the data dictionary must provide information about its logical and physical characteristics.

> In the case of replicated data, where functionally identical copies of the data are stored at multiple nodes in the network, the distributed DD/DS [data dictionary] must have knowledge of the known redundancies throughout the network. Synchronization of updates in this case is critical. [Ref. 33]

In a partitioned database, where only certain portions of the database are located at individual nodes, the data dictionary's role becomes even more important because "it must know the relationships among the pieces, and be able to manage all the parts, such that this physical dispersion of the data is transparent to the user" [Ref. 34]. Finally, the distributed data dictionary may be required to perform transformation of data to support various users. If serving a heterogeneous network--one in which dissimilar types of hardware and software coexist--the data dictionary will have to translate between different data and storage structures.

> The distributed DD/DS [data dictionary] can facilitate these translation processes by providing the metadata mappings to allow the source to be transformed into the target data. This is accomplished by storing in the data dictionary the source and target metadata descriptions to be used by the mapping process. [Ref. 35]

It is possible for the distribution of dictionary capabilities to be accomplished by several alternative configurations. One possible configuration, as mentioned earlier, involves duplicating the data dictionary in its entirety at each node of the network. An example of this is shown as Figure 6.1. (Dashed lines indicate node-to-node communications and dotted lines indicate dictionary-to-dictionary

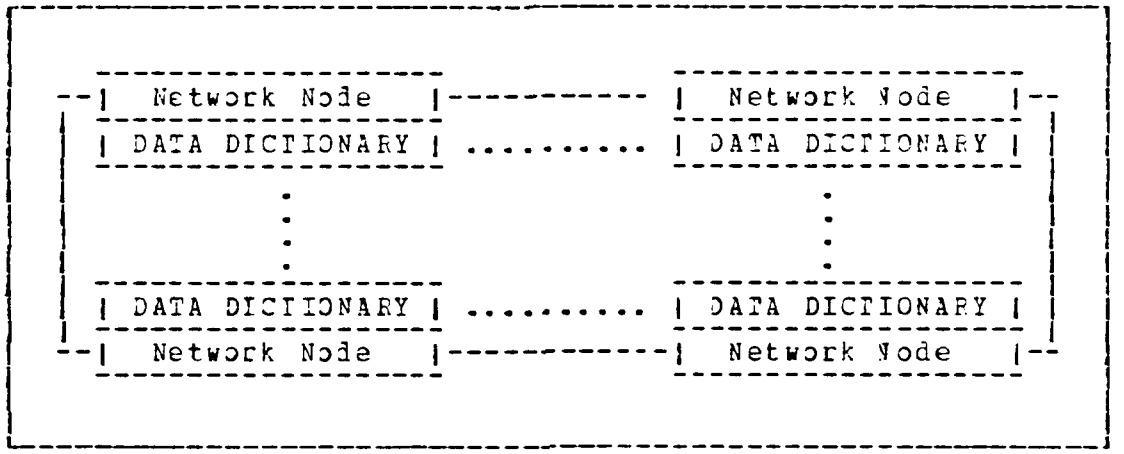


Figure 6.1 Duplicated Data Dictionaries

communications.) Each data dictionary will contain a complete copy of the entire organization's metadata. While the nodes themselves will interact frequently, the various copies of the dictionary will not. However, when one copy of the dictionary is updated, all other copies must be automatically updated if data integrity is to be maintained. This duplication of metadata will result in some degree of additional overhead, but it will improve the responsiveness of the system and minimize the necessity of inter-data dictionary queries. In some implementations, communication costs can be significantly reduced. This configuration will be most desirable in cases in which the organization's database(s) are also duplicated at each node or if nodes would

be likely to access each other's metadata often. A stable organization with well-established data processing, where metadata is not continuously being updated, would benefit most from this configuration.

In the second configuration, the data dictionary is partitioned among the various network nodes. As shown in Figure 6.2, each node contains only that portion of the dictionary that contains the metadata it requires. No one

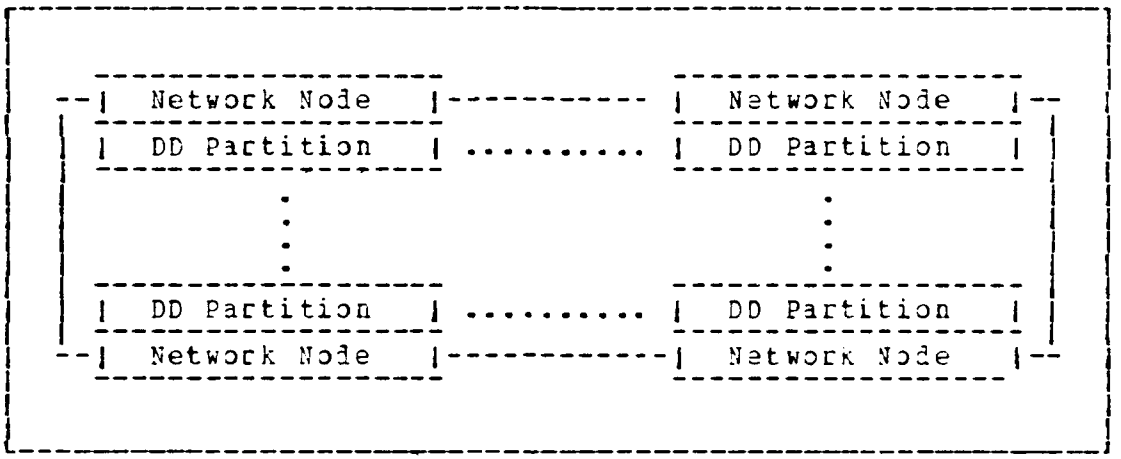


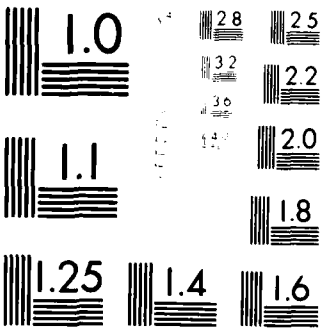
Figure 6.2 Partitioned Data Dictionary (DD)

node or station within the system will have a complete data dictionary. This configuration would be used when there is not much need for the nodes of the network to access each other's metadata and there is a relatively clear-cut differentiation between the functions being carried on at each node, which implies different metadata. Because redundancy is kept to an absolute minimum, problems could arise if a node's data dictionary partition were lost unless good backup procedures were in effect. Since each node is only responsible for maintaining its own portion of the whole, there is little update overhead and thus little system delay as long as the required metadata exists at that particular node.

AD-A152 134 AN ANALYSIS OF DATA DICTIONARIES AND THEIR ROLE IN INFORMATION RESOURCE MANAGEMENT(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA S L LANDIN ET AL. SEP 84 2/2

UNCLASSIFIED F/G 5/2 NL

END



MICROCOPY RESOLUTION TEST CHART

In the final configuration, the data dictionary is distributed in a hierarchical structure. There will be one "master" copy of the dictionary and one or more partial copies throughout the network, as shown in Figure 6.3. In this configuration, each node that contains a portion of the

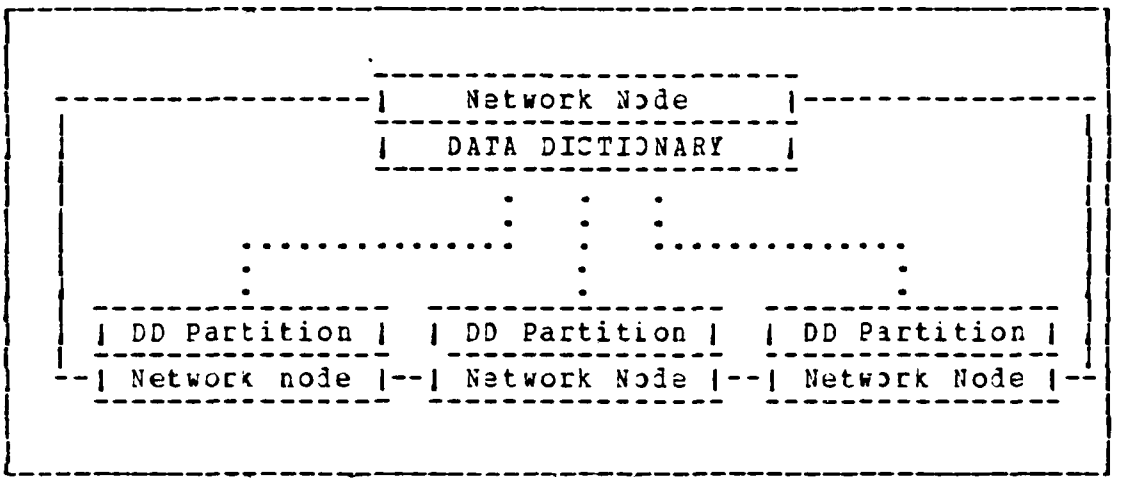


Figure 6.3 Hierarchy of Distributed Data Dictionaries

data dictionary is responsible for updating the master dictionary whenever its portion is modified. This structure ensures data integrity and provides flexibility by allowing varying amounts of metadata to be distributed. Another use for this hierarchical structure might be to separate functionality within a network, e.g., database, automated office, and programming functions. Each of these functions is able to maintain its portion of the dictionary locally while one master copy is available to handle inter-partition queries.

There are presently several commerical packages in the development or testing stages that will be able to satisfy the requirements of distributed processing. One system that is already available and being used in numerous applications

is ADR's Relational Information Management Environment (RIME) system. As discussed in Chapter V, this system features fourteen separate components that can be integrated into one "total" system. One component, D-NET, combines a database, data dictionary, and communications interfaces to support the special requirements of distributed processing. D-NET is capable of supporting both homogeneous and heterogeneous networks:

> The flexibility provided by D-NET and the other software components allows users to configure the distributed system networks based on the needs of each node. Various operating systems, computer types, and cooperating software products can be used to create a specific environment without impacting application development and operations. [Ref. 36]

D-NET can implement the system's data dictionary, DATADICTIONARY, as either one centralized dictionary or as multiple copies stored at remote locations. Similarly, RIME's database, DATACOM/DB, can be maintained either at one centralized location or distributed to various nodes throughout the network. D-NET serves as the basis of the Army's project VIABLE, providing numerous benefits that include cost effectiveness, highly expandable, increased productivity, resource control and synchronization, and independent operation at the local user's level.

B. DECISION-MAKING

In this section we will show how the data dictionary provides managers with the efficiently recorded, accurate, and timely information necessary to make decisions in consonance with the goals of the organization, whether in a centralized or distributed environment. According to the report of the Committee on Review of Navy Long-Range ADP Planning, "information technology", which includes data dictionaries, is

> critical to the Navy's ability to fulfill its wartime and peacetime roles in an optimum manner. The available technologies would enable the Navy to approach its missions with information and data that (1) have been collected and recorded simply, (2) have improved accuracy, (3) have been speedily reported, collated, and distributed, (4) lead to summaries that are timely and to the point, as and when needed, and (5) have enabled both manpower commitments and costs to be reduced. [Ref. 37]

1. The Decision-Making Process

Herbert Simon's classic model of the decision-making process, as cited by Sprague and Carlson [Ref. 38], consists of three distinct steps: intelligence, design, and choice. The use of a data dictionary supports the decision maker as he takes each step.

a. Intelligence involves searching the environment for conditions calling for decisions. Raw data must be obtained, processed, and examined for clues that may identify problems. However, so much data is available within an organization that a seemingly infinite parade of information can be produced--this situation is called information overload. There must be some way of narrowing down the amount of information that is presented to the decision maker. A data dictionary used in conjunction with a database can play an important role in this narrowing process. As discussed earlier in the thesis, the dictionary helps an organization identify and eliminate redundant data. Its query language can be used to select infomation about a particular entity and its report definition capability can be used to generate aggregate, rather than detailed data. Relationships betwen entities are easily identified so that managers' questions such as "What is the range of values for 'Readiness Status' data?" and "Which departments receive the 'Ammunition Transaction' report?" can be answered.

b. Design entails inventing, developing, and analyzing possible courses of action. This involves processes to understand the problem, to generate solutions, and to test solutions for feasibility. The data dictionary plays a key role in documenting the decision maker's environment so that he or she will have a centralized source of information from which to develop possible choices. The dictionary can also be used to tailor information to meet specific needs by defining user views of data and restricting user access to certain data. In this way, users can be presented only with the information they are supposed to have and need to have, as determined by higher authority in the organization, instead of having to deal with non-essential information.

In addition to recording information about the plans, structure, and functions of the organization, the data dictionary can also be used to record information about the decision makers themselves. In the case of the U.S.S. Constellation, for example, information about the commanding officer and the key elements of his environment can be documented: which decisions he wishes to make and which ones his subordinates will make, the mission assigned to the carrier by the C.O.'s superiors, the relative priorities he attaches to various subjects, his short term and long term personal goals, previous decisions he has made, and so on.

c. Choice involves selecting a particular course of action from those available and implementing that choice. Of course, the ultimate decision will lie with the decision maker, and not with the data dictionary. At best, the data dictionary can present options to the decision maker and, once the choice is made, can document the steps taken to implement that choice.

2. Crisis Management

The accuracy and timeliness of information provided to the decision maker becomes of critical importance when the decision-making process occurs during a crisis situation. In wartime, for example, there is usually a great deal of risk associated with a decision: many decision makers are involved, information must be consolidated from a variety of sources and locations, little time is available to make decisions, and, due to the uniqueness of events, there is often no pre-defined structure for making the decision. There are four ways that the data dictionary can prove especially helpful in crisis decision-making.

a. The dictionary speeds up the information-gathering process. As discussed earlier, user views and accesses have been pre-defined and can be changed easily as needed. Active data dictionaries provide for automatic update of any changes that are made, so information is always current.

b. The dictionary prioritizes information. The priorities of the organization and the decision makers are taken into account and can be updated as events occur. In this way, the attention of decision makers is focused on truly important information rather than dispersed over a wide range of information.

c. The dictionary provides a common information base. This is important when many decision makers at different locations are involved. All participants have the latest information and can also take advantage of the "corporate memory" provided by the dictionary.

d. In short, the dictionary provides 'intelligent" information management. It reduces information overload, tailors information to specific decision-makers' needs, and responds well to infrequent, ad hoc requests. It helps to establish relationships between events as they occur.

The typical, or even "ideal", data dictionary will not be able to fully support the decision-making process without the help of additional sophisticated software to take advantage of its capabilities. We believe that as the acceptance and use of the data dictionary as a tool for information resource management become widespread, the demand for an expanded role for the dictionary will increase. Organizations must become more accomplished in the top-down planning process of the system development life cycle in order to receive maximum benefits from data dictionary technology.

C. CONCLUSIONS

In this thesis, we have discussed the structure, functions, and objectives of a data dictionary. We have compared popular commercial products to an "ideal" dictionary based on criteria we developed and on FIPS DDS guidelines. We have analyzed the role of a data dictionary in information resource management, including its support of a distributed data processing environment and of the decision-making process. It seems clear that as organizations become cognizant of the need to manage their information efficiently, the importance and necessity of data dictionary implementation will continue to increase.

Designers of data dictionaries are aware of these trends and are moving in the following directions:

> First, toward what is known as an integrated data dictionary and second, toward a free-standing dictionary that serves as a driver of a distributed data processing system made up of several types of computers, data base management systems, file managers, and text editors. [Ref. 39]

In reference to the first projection, several commercial systems have been developed that feature integration of a

data dictionary with a database. One example of this is ADR's RIME which features integration of a database and a data dictionary with numerous other components to form one very capable and flexible system. Addressing the second projection, Rullo [Ref. 40] foresees development of a "super data dictionary" to support future integrated and distributed systems:

> In this environment, the data dictionary would act as a driver of the system. The data dictionary/data directory might also have some integrated facilities permitting transfer of data among other system software functions including itself. There is a trend in this direction, with other systems depending on the data dictionary/data directory and that system itself beginning to resemble a model of the enterprise.

We believe the future holds significant improvements and expansions of data dictionary technology. It is important that the development of standards for data dictionary compatibility continue along with the development of standards that are currently being developed to support network communications. It is conceivable that these standards, if widely accepted, would allow any data dictionary to "talk" to another and to exchange information. The FIPS DDS standards developed by the National Bureau of Standards will most likely become the basis for data dictionaries procured and used by the federal government.

We also foresee the use of fourth generation languages, the extremely user-friendly, "close-to-natural-language" languages that will facilitate user access to the dictionary's metadata. These languages will replace the formal command languages and awkward syntax described earlier in the thesis. Another factor contributing to the increased utility of data dictionaries will be the use of sophisticated software and artificial intelligence techniques in conjunction with the dictionary. As the central source of

data about an organization, the data dictionary contains a broad base of information upon which an artificial intelligence "expert" system can be built. For example, it is possible that an expert system would be able to verify and validate additions to the dictionary schema based on predetermined rules and information gained from previous manipulations of the schema. It would also be able to establish associations between the contents of the data dictionary and flag them for the attention of the decision maker. In addition, a "smart" data dictionary would be able to "realize" that every time a user logs on to the system, he asks for particular information, so that eventually, the data dictionary will provide it for him automatically.

No matter what changes occur in data dictionary technology, the data dictionary's role in the efficient management of an organization's information resource will continue to be an increasingly important one. The dictionary will support the organization in its planning and analysis of functions, its development of information systems, the maintenance of those systems, and the intelligent use of those systems. We believe that the military will soon provide a vast market for data dictionary software and that the demands of its users will drive data dictionary technology even further.

APPENDIX A
BACKUS-NAUR FORM

Backus-Naur form is a graphic notation for describing the syntax of a language. It is used by the Federal Information Processing Standard for Data Dictionary Systems (FIPS DDS) to show the format of the commands used to manipulate the dictionary. The following are common Backus-Naur symbols used by the FIPS DDS:

< > denotes a word or phrase

| indicates a choice between two or more alternatives, "or"

[ ] represents an option that the user may or may not include

{ } is used to set off choices separated by "|" and to enclose the format of the command

The syntax for the ADD-ENTITY command appears as follows:

```
ADD-ENTITY
    {[OF] {ENTITY-TYPE | E-T} <entity-type-name>
    WHERE NAME [IS] <name-clause>
    [WHERE {ATTRIBUTE | A} [FOR] <attribute-clause-1>
       [,..., attribute-clause-n]]]
    WITH SECURITY <security-clause>]}
```

It indicates that there are several different ways of adding an entity to the dictionary. At a minimum, the command must include ENTITY-TYPE or E-T, an entity-type name, WHERE NAME, and a name clause. The words OF and IS are optional, as are the last two phrases set off by brackets. If the phrases are used, the same rules hold for choosing elements within them.

LIST OF REFERENCES

1. Kroenke, David M., Database Processing, p. 1, Science Research Associates, 1983

2. Committee on Review of Navy Long-Range ADP Planning, "Navy Nontactical Automatic Data Processing Policy, Organization and Management--Interim Report to the United States Navy," p. 2, July 1983

3. Durrell, William, "Disorder to Discipline Via the Data Dictionary," Journal of Systems Management, v. 34, p. 19, May 1983

4. Applied Data Research, Inc., Introduction to DATADICTIONARY, 1982

5. Lefkovits, Henry C., Sibley, Edgar H., and Lefkovits, Sandra L., Information Resource/Data Dictionary Systems, p. 2-6, QED Infosciences, 1983

6. National Bureau of Standards, Federal Information Processing Standard for Data Dictionary Systems, August 1983

7. Goldfine, Alan H., ed., Data Base Directions: Information Resource Management--Strategies and Tools, National Bureau of Standards, p. 19, September 1982

8. Goldfine, p. 18

9. Leong-Hong, Belkis W. and Plagman, Bernard K., Data Dictionary/Directory Systems: Administration, Implementation, and Usage, pp. 25-58, John Wiley & Sons, 1982

10. Leong-Hong and Plagman, p. 32

11. Leong-Hong and Plagman, p. 45

12. Allen, Frank W., Loomis, Mary E. S., and Mannino, Michael V., "The Integrated Dictionary/Directory System," Association for Computing Machinery Computing Surveys, v. 14, pp. 246-247, June 1982

13. Leong-Hong and Plagman, p. 224

14. Van Duyn, J. A., Developing a Data Dictionary System, p. 39, 1982

15. Van Duyn, p. 40

16. Schussell, George, "The Role of the Data Dictionary," Datamation, v. 23, p. 133, June 1977

17. Kreitzer, Lawrence W., "Data Dictionaries--The Heart of IRM," Infosystems, v. 28, p. 64, March 1981

18. Van Duyn, pp. 39-40

19. Leong-Hong and Plagman, p. 132

20. Database Design, Inc., Data Designer User Guide, pp. 3-4, 1983

21. MSP, Inc., DATAMANAGER User's Guide, pp. 1-2, 1983

22. MSP, Inc., pp. 2-6

23. Goetz, Martin, "The ADR Integrated System", Infosystems, v. 29, p. 4, September 1982.

24. Lefkovits, Sibley, and Lefkovits, p. 6-64

25. Kroenke, p. 117

26. Lefkovits, Sibley, and Lefkovits, p. 1-45

27. Lefkovits, Sibley, and Lefkovits, p. 1-47

28. Allen, Loomis, and Mannino, p. 266

29. Leong-Hong and Plagman, p. 229

30. Committee on Review of Navy Long-Range ADP Planning, p. 9

31. Van Rensselaer, Cort, "Centralize? Decentralize? Distribute?", DATAMATION, v. 25, p. 97, April 1979

32. Committee on Review of Navy Long-Range ADP Planning, p. 13

33. Leong-Hong and Plagman, p. 234

34. Leong-Hong and Plagman, p. 234

35. Leong-Hong and Plagman, p. 236

36. Applied Data Research, Inc., ADR/D-NEI: Concepts and Facilities, p. 3, 1982

37. Committee on Review of Navy Long-Range ADP Planning, p. 3

38. Sprague, Ralph H., Jr., and Carlson, Eric D., Building Effective Decision Support Systems, pp. 26-27, Prentice Hall, 1982

39. Bullo, Thomas A., Advances in Data Base Management, p. 138, Heyden & Son, Inc., 1983

40. Bullo, p. 139

INITIAL DISTRIBUTION LIST

		No. Copies
1.	LT Suzanne L. Landin 7957 De Arment Court Springfield, Virginia 22153	3
2.	LCDR Ronald L. Owens 15161 Hollyside Drive Dumfries, Virginia 22026	6
3.	Professor Daniel R. Dolk, Code 54Dk Naval Postgraduate School Monterey, California 93943	5
4.	MAJ F. Marchman Perry, Code 55Pj Naval Postgraduate School Monterey, California 93943	1
5.	Computer Technology Programs, Code 37 Naval Postgraduate School Monterey, California 93943	1
6.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
7.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2

END

FILMED

4-85

DTIC